НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 «Програмна інженерія»

на тему: «Веб-сервіс для графічного представлення аналітичних звітів»

Виконав:
студент IV курсу, групи КП-52
Піщела Олександр Костянтинович _____

Керівник:
Старший викладач кафедри ПЗКС, к.т.н.,
Хіцко Я.В. _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н.,
Онай М.В. _____

Рецензент:
Доцент кафедри СПСКС ФПМ, к.т.н.,
Боярінова Ю.Є. _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2019 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки – 6.050103 «Програмна інженерія»

ЗАТВЕРДЖУЮ
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2018 р.

ЗАВДАННЯ
на дипломний проект студенту

Піщелі Олександру Костянтиновичу

1. Тема проекту «Веб-сервіс для графічного представлення аналітичних звітів», керівник проекту Хіцко Яна Володимірівна, к.т.н., доцент, затверджені наказом по університету від «22» травня 2019 р. № 1331-С

2. Термін подання студентом проекту «16» червня 2019 р.

3. Вихідні дані до проекту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - аналіз існуючих програмних рішень;
   - обгрунтування вибору засобів реалізації;
   - розроблення архітектури та програмних модулів;
   - аналіз розробленого програмного забезпечення.

5. Перелік обов'язкового графічного матеріалу:
   - діаграма послідовності (креслення);
   - діаграма діяльності (креслення);
   - діаграма компонентів (плакат);
   - діаграма прецендентів (плакат).

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2018 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Вивчення літератури за тематикою проекту	14.11.2018	
2.	Розроблення та узгодження технічного завдання	28.11.2018	
3.	Розроблення структури веб-сервісу	15.12.2018	
4.	Підготовка матеріалів першого розділу дипломного проекту	30.12.2018	
5.	Розроблення дизайну сторінок та графічних елементів	03.02.2019	
6.	Підготовка матеріалів другого розділу дипломного проекту	20.02.2019	
7.	Програмна реалізація веб-сервісу	10.03.2019	
8.	Тестування веб-сервісу	17.03.2019	
9.	Підготовка матеріалів третього розділу дипломного проекту	30.03.2019	
10.	Підготовка матеріалів четвертого розділу дипломного проекту	11.04.2019	
11.	Підготовка графічної частини дипломного проекту	21.04.2019	
12.	Оформлення документації дипломного проекту	26.05.2019	

Студент О.К. Піщела

Керівник проекту Я.В. Хіцко

ВЕБ-СЕРВІС ДЛЯ ГРАФІЧНОГО ПРЕДСТАВЛЕННЯ АНАЛІТИЧНИХ ЗВІТІВ

АНОТАЦІЯ

Даний дипломний проект присвячений розробленню веб-сервісу для графічного представлення аналітичних звітів.

Розроблене програмне забезпечення являє собою сайт, який надає змогу завантажувати на сайт готовий аналітичний звіт, зберігати звіти в обліковому записі користувача, автоматично генерувати графіки з цих звітів, міняти типи графіків та динамічно змінювати дані на графіках .

У даному дипломному проекті було розроблено архітектуру, модуль авторизації та реєстрації, модуль графіків, модуль завантаження файлів та модуль зв’язку з БД, також розроблено дизайн проекту. Для збільшення продуктивності клієнтської частини, було використано серверний рендеринг.

WEB SERVICE FOR GRAPHICAL REPRESENTATION OF ANALYTICAL REPORTS

ABSTRACT

This graduation project is devoted to the development of a web service for graphical representation of analytical reports.

The developed software is a site that allows you to download a ready-made analytical report to a site, save reports in a user's account, automatically generate charts from these reports, change chart types, and dynamically change data on charts.

In this diploma project, an architecture, authorization and registration module, a graphics module, a file upload module, and a module for communicating with the database were developed, and a design of project was developed. To increase the client's performance, server rendering was used.

ДП.045440-01-90 Веб-сервіс для графічного представлення аналітичних звітів.
Відомість проекту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проекту		
ДП.045440-02-91	Веб-сервіс для графічного	4	
	представлення		
	аналітичних звітів		
	Технічне завдання		
ДП.045440-03-81	Веб-сервіс для графічного	59	
	представлення		
	аналітичних звітів.		
	Пояснювальна записка		
ДП.045440-04-51	Веб-сервіс для графічного	4	
	представлення		
	аналітичних звітів.		
	Програма та методика		
	тестування		
ДП.045440-05-34	Веб-сервіс для графічного	8	
	представлення		
	аналітичних звітів.		
	Керівництво користувача		
ДП.045440-06-99	Веб-сервіс для графічного	1	
	представлення		
	аналітичинх звітів.		
	Зв’язок модулів. Діаграма		
	компонентів		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-07-99	Веб-сервіс для графічного	1	
	представлення		
	аналітичинх звітів.		
	Алгоритм створення		
	графічного		
	представлення звіту		
	Діаграма діяльності		
ДП.045440-08-98	Веб-сервіс для графічного	1	
	представлення		
	аналітичинх звітів.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

ВЕБ-СЕРВІС ДЛЯ ГРАФІЧНОГО ПРЕДСТАВЛЕННЯ АНАЛІТИЧНИХ ЗВІТІВ

Технічне завдання

ДП.045440-02-91

"ПОГОДЖЕНО"

Керівник проекту:

_____ Я.В. Хіцко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ О.К. Піщела

2018

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Веб-сервіс для графічного представлення аналітичних звітів.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проектування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання в якості веб-сервісу з метою надання необхідних функцій для графічного представлення аналітичних звітів.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Веб-сервіс повинен забезпечувати такі основні функції:

1) можливість завантаження на сервіс звітів;
2) можливість генерування графіків із завантажених звітів;
3) можливість авторизації, реєстрації;
4) можливість перегляду своїх звітів;
5) можливість динамічного оновлення даних на графіках;
6) можливість змінювати типи графіків;
7) можливість збереження створеного графічного представлення у різних форматах;
8) можливість динамічно змінювати дані на графіках.

Розробку виконати на платформі Microsoft .Net з використанням технології ASP.NET AJAX.

Додаткові вимоги:

1) наявність динамічного меню;
2) наявність анімованих кнопок;
3) використання логотипу Центру електронної освіти у якості посилання на головну сторінку web-сайту Центру;
4) наявність на web-сторінках місця для розміщення логотипів партнерських організацій;
5) дизайн сторінок з використанням в якості базових білого та синьо-зеленого кольорів.

5. ВИМОГИ ДО ПРОЕКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проекту повинна бути розроблена наступна документація:

1) пояснювальна записка;
2) програма та методика тестування;
3) керівництво користувача;
4) креслення:
   - «Авторизація користувача. Схема алгоритму»;
   - «Синхронізація даних. Схема роботи програмної системи».

6. ЕТАПИ ПРОЕКТУВАННЯ

Вивчення літератури за тематикою роботи……..14.11.2017

Розроблення та узгодження технічного завдання……....28.11.2017

Розроблення структури web-ресурсу…….....15.12.2017

Розроблення дизайну сторінок та графічних елементів……..03.02.2018

Програмна реалізація web-ресурсу……....17.03.2018

Тестування web-ресурсу…….....03.04.2018

Підготовка матеріалів текстової частини проекту……..….28.04.2018

Підготовка матеріалів графічної частини проекту……...…12.05.2018

Оформлення технічної документації проекту……..…25.05.2018

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ВЕБ-СЕРВІС ДЛЯ ГРАФІЧНОГО ПРЕДСТАВЛЕННЯ АНАЛІТИЧИНХ ЗВІТІВ

Пояснювальна записка

ДП.045440-03-81

"ПОГОДЖЕНО"

Керівник проекту:

_____ Я.В. Хіцко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ О.К. Піщела

2019

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення

БД – база даних

СКБД – система керування базою даних

ОС – операційна система

Front end − це інтерфейс для взаємодії між користувачем і back end

Back end − програмно-технічна частина сервісу

UI (User interface) – графічний інтерфейс користувача, який забезпечує передачу інформації між користувачем і програмно–апаратним компонентом комп'ютерної системи.

SQL (від англ. Structured query language) – декларативна мова програмування для взаємодії користувача з базами даних).

NoSQL (від англ. no only SQL) – база даних, що забезпечує інший механізм зберігання та видобування даних, ніж звичайний підхід таблиць-відношень в реляційних базах даних.

JVM (від англ. Java Virtual Machine) – віртуальна машина Java – основна частина виконуючої системи Java, під назвою “Java Runtime Environment” (JRE).

MVC (з англ. Model View Controller) – схема розділення даних додатку.

API (від англ. Application Programming Interface) – прикладний програмний інтерфейс, набір визначень взаємодії різнотипного програмного забезпечення).

Фреймворк – інфраструктура програмних рішень, що полегшує розробку складних систем.

SDK (від англ. Software Development Kit) – набір засобів розробки, що дозволяє спеціалістам з програмного забезпечення створювати додатки для визначеного пакету програм.

SSR (з англ. Server-Side Rendering) – рендеринг на сервері клієнтської частини або універсального застосування в HTML.

SVG (від англ. Scalable Vector Graphics) – мова розмітки масштабної векторної графіки.

JSON (JavaScript Object Notation) – формат обміну даними між комп'ютерами.

HTTP (від англ. Hypertext Transfer Protocol) – протокол передачі гіпер-текстових документів, що використовується в комп'ютерних мережах.

HTML (від англ. HyperText Markup Language) – стандартна мова розмітки для створення веб-сторінок і веб-додатків.

OAuth (скорочення від англ. Open Authorization) – це відкритий стандарт авторизації, який дозволяє користувачам відкривати доступ до своїх приватних даних (фотографій, відео, списки контактів), що зберігаються на одному сайті, іншому сайту, без необхідності вводу імені користувача або паролю.

PNG (від англ. Portable Network Graphics) – растровий формат збереження графічної інформації, що використовує стиснення без втрат.

3D (від англ. 3-dimensional) – може означати що-небудь, що має три виміри: тривимірний простір, тривимірна графіка, тощо.

ACID (Atomicity, Consistency, Isolation, Durability) – атомарність, узгодженість, ізольованість, довговічність, це набір властивостей, що гарантують надійну роботу транзакцій бази даних.

JPEG (від англ. Joint Photographic Experts Group) – растровий формат збереження графічної інформації, що використовує стиснення з втратами.

PDF (від англ. Portable Document Format) – відкритий формат файлу для представлення двовимірних документів у незалежному від пристрою виведення та роздільної здатності вигляд.

CSV (від англ. Comma-Separated Values) – текстовий формат, призначений для представлення табличних даних.

MVC (від англ. Model-View-Controller) – схема поділу даних програми, призначеного для користувацького інтерфейсу і керуючої логіки на три окремих компонента: модель, уявлення і контролер.

ВСТУП

Основна мета дипломної роботи – розроблення програмного забезпечення сервісу, з допомогою якого будь-яка звичайна людина зможе створити графік з потрібних даних для використання в Інтернеті, презентаціях та доповідях. При цьому йому не знадобиться встановлювати ніякі додаткові програмні засоби. Все, що необхідно – браузер та з'єднання з мережею Інтернет. Підставою для розробки даного веб-сервісу стало надати змогу користувачам створювати графіки в пару кліків миші не використовуючи додаткове ПЗ, та не маючи потрібних навичок.

Сьогодні люди стикаються з побудовою графіків де завгодно :в школі, університеті, на роботі, тощо.

Найчастіше користувач, коли в нього виникає потреба в побудові графіків, починає шукати безкоштовне ПЗ та витрачає багато часу на його пошуки, установку та взаємодію з UI, саме тому виникає потреба в загальнодоступному та простому сервісі, який буде вирішувати цю задачу.

Програмне забезпечення цього дипломного проекту надає наступні можливості:

– зручний UI;
– генерування різних типів графіків;
– зміна кольорів на графіку, надписів, тощо;
– зберігання своїх звітів та графіків у своєму обліковому запису, та швидке маніпулювання ними;
– редагування завантаженого звіту;
– динамічне редагування даних на графіку;
– авторизація на веб-сервіс завдяки соціальним мережам;
– користування веб-сервісом без реєстрації (у режимі анонімного користувача), це зроблено для того, щоб користувач, який більше не буде мати потреби в даному додатку, або користувач, який не хоче

зберігати свої дані в обліковому запису, не марнував час на реєстрацію;

– перегляд створених графіків на форумі, та їх оцінка.

1. АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

1.1. Загальний опис проблеми

Головна проблема – компактне представлення інформації. Якщо користувачу потрібно компактно представити великий масив будь-якої інформації, то йому потрібно спробувати скористатися для цього прийомами інфографіки. Зараз у інфографіки дуже широкий спектр застосування – її використовують в статистиці, журналістиці, рекламі, бізнесі, сфері освіти, тощо. Простими словами інфографіка це будь-яке об’єднання тексту та графічних символів з метою організації даних, демонстрації фактів, тенденцій, залежностей. Є кілька стадій розробки макету інформаційної графіки: збір, перевірка та аналіз даних, постановка цілей і виявлення патернів, кодування даних за допомогою візуальних символів. Потім наступає ключовий момент: готовий продукт пропонується аудиторії, і важливий момент на цьому етапі – як аудиторія сприйме інфографіку – чи вдасться передати людям основну мету графіки і чи зможуть люди розшифрувати візуальні символи (що означає кожен окремо взятий елемент в роботі). Саме поняття інфографіки – це частина комунікативного дизайну, в фундаменті якого лежить графічне подання зв'язків, інформації числових даних та знань. Збільшення щільності даних. Людське око здатне розрізняти величезна кількість окремих елементів. Кількість точок, розташованих на якийсь площі, тобто щільність даних потрібно намагатися максимізувати. До жаль, часто інфографіку намагаються вибудувати з двох-трьох змінних. Потрібно намагатися використовувати більшу кількість елементів, тому що така графіка цікава і притягує увагу, її цікаво розглядати і вивчати як окремі деталі, так і в цілому.

Психологами усього світу доведено, що людина краще сприймає і осмислює ту інформацію, яку він може побачити. Це не дивно, адже саме завдяки зору ми отримуємо 70-80% всіх даних про навколишній світ.

Інфографіка за своєю суттю є красивою блок-схемою, яка містить крім тексту ще й якесь візуальне оформлення. Таким чином, з одного боку, ми можемо розглядати її як з чисто утилітарною боку (надання будь-якої інформації), так і з естетичної (продуманий дизайн).

Основними перевагами інфографіки є:

– наочність представлення інформації;

– узагальненість поданої інформації;

– економія часу і місця для демонстрації великого обсягу даних;

– барвистість і привабливість для кінцевого користувача.

Єдиним же недоліком можна назвати, хіба що, поверховість наданих даних і відсутність стимулу їх додаткового осмислення [1].

Нижче наведені основні помилки при створенні макета інфографіки:

– Створення інфографіки з нічого. Коли величезна площа макета зайнята непотрібними зображеннями, то дані не помічаються, тому що їх просто недостатньо для створення візуалізації. Є два шляхи вирішення цієї проблеми: міняти спосіб подачі даних чи шукати більш повні дані. Міняти спосіб подання інформації можно з інфографіки на таблицю або ж у текстовий формат.

– Відсутність легенд і пояснень до інфографіки. Відсутність підписів до осей графіка. Якщо не пояснювати кодування візуальних символів, використаних в інфографіці, то слухачу неможливо зрозуміти, що зображено на ній. Часто автори інфографіки забувають прикріпити легенду чи підписати вісь, але це займає небагато часу.

– Повсюдне невірне застосування кругових діаграм в інфографіці. Кругова діаграма – досить зручний і наочний інструмент для того, щоб показати частку від цілого. Для того, щоб вони добре зчитувалися необхідно щоб, сума всіх представлених показників дорівнювала 100%; кількість секторів має бути менше шести, в іншому випадку порівняти сектори між собою буде важко.

Найпоширеніші помилки – використовувати кругові діаграми там, де їх використання недоречно, ділити коло на більшу кількість секторів, ніж шість і не пропорційно показувати розмір частки від цілого.

– Достаток тексту в інфографіці. Інфографіка – це перш за всього графічне відображення інформації. пояснення мають право на існування і повинні бути присутніми в макеті, але, часто вони займають всю площу макета. якщо існує проблема нестачі даних для створення інфографіки, краще обмежиться текстом;
– відсутність посилання на джерело інформації [2].

1.2. Аналіз існуючих програмних рішень

Провівши аналіз різних джерел інформації, а також існуючих програмних рішень, які пов’язані з темою графічного представлення аналітичних звітів, було виділено декілька основних ПЗ, які представлені нижче.

1.2.1. *Piktochart – веб-сервіс для інфографіки*

Переваги Piktochart:

– функціональність безкоштовної версії набагато більше ніж в його аналогів;
– готовий шаблон можна редагувати після попереднього перегляду;
– створену інфографіку можна не тільки розмістити в соціальних мережах або на сайті, але і безкоштовно завантажити.

Недоліки:

– безкоштовно в користуванні всього декілька шаблонів;
– не адаптований для мобільних пристроїв;
– обмеження на завантаження файлів – одному користувачеві видається близько 60 МБ. Розширити обсяг сховища можна тільки покупкою преміум-підпискою;

– тільки англомовна версія веб-сервісу.

1.2.2. *Infogr.am – веб-сервіс для інфографіки*

Функціональний потенціал безкоштовної та платної версії вражає: різнобарвні діаграми, статистика Google Analytics, тощо. При цьому сервіс ідеально підходить для роботи як з невеликими текстовими блоками, так і величезними масивами даних, які можна вивантажувати, наприклад з Excel чи Google Drive.

Переваги:

– багато варіацій індивідуалізації шаблонів і велика кількість інструментів для презентації даних;
– легкий у користуванні інтерфейс, який покращують корисні спливаючі підказки;
– якщо виконуючи інфографіку ви передумали та вирішили змінити тему, то внесені вами дані автоматично збережуться;
– у безкоштовної версії є можливість аналітики на той випадок, якщо вам буде цікаво дізнатися, скільки людей переглянуло ваші роботи і які саме. Приємними доповненнями, також, є онлайн-підтримка та наявність власного співтовариства, та блогу (все-таки допомога і приклади більш досвідчених користувачів ніколи не завадять).

Недоліки:

– Найбільший недолік – необхідність придбати повну версію облікового запису , щоб отримати можливість завантажувати свої роботи ($ 15 в місяць) [3].

1.3. Постановка задачі

В результаті проведеного дослідження можна сформулювати такі вимоги до програмного забезпечення веб-сервісу для графічного представлення аналітичних звітів:

– веб-сервіс має бути адаптований для мобільних пристроїв;

– можливість генерувати різні типи графіків;
– максимально простий і зручний UI, так як у користувачів не буде багато часу на його ознайомлення;
– можливість редагування завантаженого звіту;
– можливість зробити графічне представлення звіту без реєстрації на сайті;
– можливість авторизації на сайті використовуючи соціальні мережи;
– архітектурний модуль, який стане основою зв’язку інших модулів та забезпечить масштабування ПЗ;
– сервер або сервіс для зберігання звітів поза межами веб-сервісу;
– у користувача має бути досить великий обсяг віддаленого сховища;
– можливість одразу загружати кілька звітів на веб-сервіс;
– наявність форуму, де користувачі будуть мати можливість поділитися своїми графічними представленнями аналітичних звітів, та оцінити їх;
– мовний модуль.

1.4. Висновки до розділу

У даному розділі описано особливості створення інфографіки, проаналізовано існуючі веб-сервіси для графічного представлення аналітичних звітів, виявлено їх недоліки та переваги.

Основними перевагами існуючих веб-сервісів для графічного представлення аналітичних звітів є велика кількість шаблонів для створення графіків та легку у користуванні функціональність.

Основними недоліками розглянутих веб-сервісів є відсутність адаптованості для мобільних пристроїв та відсутність можливості збереження даних.

Враховуючи виділені недоліки існуючих програмних рішень сформульовано основні вимоги для розроблення веб-сервісу.

Розроблюваний веб-сервіс має забезпечувати можливість завантаження звітів користувачем, генерувати графіки та динамічно оновлювати дані на них, також надавати можливість користувачам зберігати свої дані на веб-сервісі та завантажувати зроблений графік на пристрій у різних форматах.

2. ОБГРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Вибір мови програмування

У цьому підрозділі було проаналізовано найпопопулярніші мови програмування для розроблення веб-сервісу для графічного представлення аналітичних звітів.

2.1.1. *Огляд мови програмування Java*

Java – об’єктно-орієнтована мова програмування загального призначення, на основі класів (хоча і не чисто об’єктно-орієнтована мова). Розроблена з метою мати якомога менше залежностей реалізації. Скомпільований код на Java працює на всіх платформах без перекомпіляції. Програми Java зазвичай компілюються до "байт-коду", який може працювати на будь-якій JVM, та це не залежить від базової архітектури комп'ютера. Станом на 2018 рік, Java є однією з самих популярних мов програмування, що використовуються відповідно до GitHub, особливо для клієнт-серверних веб-додатків, з 9 мільйонами користувачів [4].

Java була створена Джеймсом Гослiном у компанії Sun Microsystems (яка з тих пір був придбаний Oracle), та пізніше випущена як головний компонент платформи Java Sun Microsystems. Своєрідні та довідкові реалізації Java-компіляторів, віртуальних машин та бібліотек класів спочатку були випущені компанією Sun під власними ліцензіями. У середині 2007 року, відповідно до Java Community Process специфікацій, ліцензовано більшість технологій Java за ліцензією GNU General Public License повторно компанією Sun. Тим часом інші виробили альтернативні аналоги цих технологій Sun, такі як компілятор GNU для Java (компілятор bytecode), GNU Classpath (стандартні бібліотеки) і IcedTea-Web (браузерний додаток для аплетів) [5].

Переваги мови програмування Java:

– Об’єктно-орієнтований. Все у Java – об'єкт. Можуть бути легко розширені доповнення, так як він заснований на об’єктній моделі.
– Платформонезалежний. На відміну від інших мов, Java створювалась в незалежному від платформи байт-коді. Він поширюється мережею Інтернет та інтерпретується в JVM, на якій він досі працює.
– Дуже простий для тих, хто розуміється в об’єктно-орієнтованому програмуванні.
– Безпечний. Методи перевірки автентичності створені на шифруванні з відкритим ключем.
– Java є архітектурно-нейтральним, тобто архітектурно-нейтральні об’єкти генеруються компілятором, що робить скомпільований код виконуваним на багатьох процесорах, якщо є система Java Runtime.
– Портативний. Архітектурно-нейтральний, без залежності від реалізації аспектів специфікацій. Компілятор в Java написаний на ANSI C з чистою переносимістю , який є підмножиною POSIX.
– Усуває помилки в різних ситуаціях. Спирається на час компіляції та на перевірку помилок, та перевірку помилок під час виконання.
– Багатонитевий. Завдяки функції багатонитевості є можливість писати програми, які мають можливість виконувати велику кількість завдань одночасно. З цією конструктивною особливістю розробники можуть створювати налагоджені ітерактивні додатки.
– Інтерпретований. Байт-код переводиться одразу в машинні інструкції, ніде не зберігається, це робить процес швидкішим та більш аналітичним, так як зв’язування відбувається як додаткове з невеликою вагою процесу.
– Високопродуктивний. Завдяки Just-In-Time компілятору отримана висока продуктивність.

– Динамічний. Так, як програмування на Java вважається більш динамічним, ніж на інших С-подібних мовах, так як він призначений для адаптації до мінливих умов [6].

Розглянемо недоліки мови програмування Java:

– немає можливості оперативного доступу до змінних і методів класу;
– немає наочного візуального уявлення про структуру проекту (є недоліком тільки в великих проектах, так як в маленьких це не відчувається) [7].

2.1.2. *Огляд мови програмування Python*

Python – інтерпретована, об'єктно-орієнтована, високорівнева мова зі строгою динамічною типізацією. Динамічна семантика з динамічним зв’язуванням та високого рівня структури даних роблять її приємною для швидкої та легкої розробки програм, та приємною як засіб поєднування наявних компонентів [8].

Python був задуманий наприкінці 1980-х років як наступник мови ABC. У 1991 році компанією Python Software Foundation був опублікований код Python, та він швидко став популярним у інтернеті. Python є повністю об’єктно-орієнтовним з початку 1990-х. Python багато запозичив з С-подібних мов, також має окремі особливості функціонального програмування з мови Lisp. Дизайнерська інтуїція Гвідо разом з дружньою спільнотою користувачів, вважаються головними факторами успіху мови Python [9].

Переваги мови Python:

– Легкий та зрозумілий у читанні (слід використовувати відступи для виділення блоків).
– Чимало корисних модулів стандартного дистрибутиву(також є модуль для розробки графічного інтерфейсу).
– Можливість застосовувати в діалоговому вікні (дуже корисний режим для розв'язання легких задач та експериментування).

– Просте, та досить об’ємне середовище розробки(стандартний дистрибутив IDLE), написане на мові Python.
– Можливість редагувати стандартний код користувачам (open-source).
– Кросплатформеність. Інтерпретатор Python реалізований практично на всіх платформах та операційних системах.
– На Python дуже висока швидкість виконання програм. Це зумовлено тим, що основні його бібліотеки написані на C++ і потрібно менше часу на виконання завдань, ніж на інших високорівневих мовах. Виходячи з цього, користувачі мають змогу писати свої потрібні модулі на C чи C++ для Python.
– Написані за допомогою Python скрипти виконуються на майже всіх сучасних ОС. Така переносимість забезпечує Python застосування в самих різних областях.
– Python підходить для майже всіх рішень в області програмування, незалежно від призначення, чи то офісні програми, чи то веб-додатки, чи GUI-додатки [10].

На мою думку, єдиним недоліком є не дуже висока швидкість виконання програми, написаної на мові Python, що зумовлюється її інтерпретованістю [11].

2.1.3. *Огляд мови програмування JavaScript*

JavaScript – прототипно-орієнтована мова програмування. Вона відображає мову ECMAScript, чиїм прототипом спочатку і була. Перша варіація з'явилася ще в 1995 році і з тих пір постійно вдосконалювалася, поки не прийшла до нинішнього вигляду. Найчастіше ця мова використовується в розробці додатків і браузерах з метою надання їм інтерактивності і «жвавості» [12].

Базовою особливістю цієї мови відзначається те, що на нього вплинули інші (Python, Java та ін.) мови програмування з метою надання

максимального комфорту JavaScript і легкості в розумінні його тими користувачами, які не мають відповідної освіти і глибинних знань. JavaScript – офіційно зареєстрована торгова марка компанії Oracle.

Завдяки мові JS доступні для виконання такі функції, як: можливість змінювати сторінки Web-браузерів, добавляти чи видаляти теги, зміни стилів сторінок, інформація про дії на сторінці, запит доступу до випадкової частини сирцевого коду сторінки, та внесення змін в цей код, виконання дій з cookie-файлами. Область застосування цієї мови дуже широка і не обмежується: програмами, які використовують JS, присутній і тестовий редактор, і додатки (як для комп'ютерів, так і мобільних і навіть серверних), і прикладне ПЗ [13].

Переваги JavaScript:

– Жоден сучасний Web-браузер не існує без підтримки JavaScript.
– Корисні функціональні налаштування.
– Глибока інтеграція з HTML/CSS, завдяки цьому JS, без заперечень, дуже важливий у веб-розробці.
– Взаємодія з застосуванням може здійснюватися навіть через звичайні текстові редактори, такі як Microsoft Office чи Open Office.
– Навіть при великій кількості не критичних помилок, код буде продовжувати свою роботу.
– У разі необхідності звільнення пам’яті з цим легко справляється Garbage Collector [14].

Недоліки JavaScript:

– Через вільний доступ до сирцевого коду популярних скриптів занижений рівень безпеки.
– Велика кількість дрібних, подразливих помилок на всіх етапах роботи та велика частина з них легко виправляється, але через них мову можна вважати менш професійною, порівняно з іншими.
– Повсюдне поширення. Специфічним недоліком можна вважати те, що частина активно використовуваних ПЗ (особливо додатків)

перестануть існувати при відсутності цієї мови, оскільки цілком базуються на ньому [15].

2.1.4 Огляд мови програмування PHP

PHP – мова, яка впроваджується у сторінку HTML. Завдяки PHP-модулю, запит перед відправкою даних клієнта проходить обробку, коли клієнт запитує цю сторінку у Web-сервера. Цей модуль міститься у Web-сервері [16]. У 1995 році був перший публічний реліз, а станом на 1997 рік PHP використовували близько 1% усіх інтернет-доменів світу [17].

Переваги PHP:

– Легка у вивченні. Використовуючи цю мову розробник швидко досягає високої продуктивності програмування. Вона дозволяє Web-розробникам легко добавляти, до своїх Web-додатків найсучасніші рішення, такі як, наприклад, динамічне генерування сторінок.
– Відкриті джерела. PHP поширюється за ліцензією, яка надає будь-який вид розробки чи використання. Виходячи з цього, є можливість використовувати програму безкоштовно. Крім того, завдяки розробникам по всьому світу, PHP постійно розвивається та покращується. Сирцевий код легко доступний, тож можна без проблем налаштувати потрібну тобі частину програми під власні потреби.
– Підтримка БД. PHP надає продуктивну підтримку БД. Без питань працює з ODBC, відкритими базами даних (MySQL і PostgreSQL), а також з комерційними (Microsoft SQL Server, Oracle і Sybase).
– Розширення. В Інтернеті користувач має змогу знайти велику кількість безкоштовно доступних програмних рішень і сирцевого коду для яких завгодно прикладних задач. Розробники мають можливість користуватися безліччю готового коду для швидкого компілювання найсучасніших додатків [18].

Недоліки PHP:

– Мова не має корисних можливостей цілком об'єктно-орієнтованої мови, як, наприклад, індивідуальні змінні та множинне спадкування, так як мова не повністю об’єктно-орієнтована.
– Відповідає корпоративним вимогам. PHP має популярність, як програма з відкритим кодом та вона технічно перевершує більшість комерційних аналогів. Однак PHP не має деяких вагомих рис, які притаманні корпоративному середовищу. З цього слідує, що не вдасться використовувати PHP у корпорації, бо буде потрібно набагато більше додаткових програмних засобів, ніж при використанні його аналогів, як наприклад, C++ [19].

2.2. Вибір системи керування бази даних

2.2.1. PostgreSQL

PostgreSQL – це вільна та відкрита системою управління реляційними базами даних (RDBMS), що підкреслює відповідність розширюваності та технічних стандартів. Він призначений для роботи з різними навантаженнями, від окремих машин до сховищ даних або веб-служб з багатьма одночасними користувачами. PostgreSQL має транзакції з властивостями атомарність , консистенції, ізоляції, довговічності (ACID), автоматично оновлюваних переглядів, матеріалізованих переглядів, тригерів, зовнішніх ключів і збережених процедур. Сьогоднішня PostgreSQL – заслуга співпраці великої кількості користувачів та компаній, які використовують цю СКБД та впроваджують у неї самі новітні досягнення. Сервер цієї СКБД написаний на мові C. Найчастіше розповсюджується, як набір текстових файлів із сирцевим кодом.

Ідея Postgres була в додаванні найменшої кількості функцій, необхідних для повної підтримки типів даних. Прототипна версія була показана у 1988 році. У 1994 замінили інтерпретатор мови запитів на SQL. По цей день проект продовжує створювати релізи, доступні за її ліцензією

PostgreSQL. Код надходить від внесків власних постачальників, компаній підтримки та програмістів з відкритим кодом [20].

Переваги PostgreSQL:

– Надійність та стабільність. На відміну від багатьох баз даних, компаніям надзвичайно часто повідомляють про те, що PostgreSQL стабільно функціонує на них протягом декількох років роботи з високою активністю.
– Розширюваність. Сирцевий код доступний для всіх безкоштовно. Якщо є потреба зробити налаштування чи розширення у будь-який спосіб, то це можна зробити з мінімальним зусиллям. Розроблення PostgreSQL відбувається за рахунок користувачів та компаній, які використовують її.
– Гнучкість, так, як в PostgreSQL об’єктно-реляційна модель даних СКБД.
– Цілісність даних. Відповідає вимогам ACID.
– Інструменти для проектування та адміністрування баз даних GUI: Існує багато високоякісних графічних інструментів, доступних для PostgreSQL як від розробників з відкритим кодом, так і від комерційних провайдерів.
– Структура даних та типи даних. PostgreSQL підтримує величезний список таких типів даних, як: XML-дані, знімок ідентифікатора транзакцій, мережеві адреси, MAC-адреси, адреси вузлів, чимало числових типів, включаючи типи з плаваючою точкою, грошові, геометричні, бінарні типи, тощо, та якщо користувачу буде його недостатньо, то він має можливість створити нові типи даних завдяки команді CREATE TYPE [21].

Недоліки PostgreSQL:

– Набагато повільніший ніж такі аналоги, як MySQL навіть при простих операціях читання.

– Не підтримує весь стандарт ANSI SQL 92 ', тим більше стандарт ANSI SQL 99'.
– Іноді є проблема з пошуком хостингу, який підтримує СКБД [22].

2.2.2. *MySQL*

MySQL – реляційна СКБД з відкритим сирцевим кодом. Це безкоштовне і відкрите ПЗ згідно з умовами ліцензії GNU General Public License, та в той же час доступна під різними власними ліцензіями. Розроблення цієї СКБД почалося у 1994 році. Спочатку вона була створена для особистого користування від mSQL на основі мови ISAM низькорівневої, яку творці вважали занадто повільними і негнучкими. Вони створили новий інтерфейс SQL, зберігаючи при цьому той же API, що і mSQL. Зберігаючи API у відповідності з системою mSQL, багато розробників змогли використовувати MySQL замість (власне ліцензованої) mSQL antecedent. Як і Java була власністю компанії Sun Microsystems. MySQL є складовою частиною стеку програмного забезпечення LAMP (та інших), що є абревіатурою для Linux, Apache, MySQL, Perl / PHP / Python. MySQL використовується багатьма веб-додатками, що керуються базами даних, включаючи Drupal, Joomla, phpBB і WordPress. MySQL також використовується багатьма популярними веб-сайтами, включаючи Facebook, Twitter, Flickr, та YouTube [23].

Переваги MySQL:

– Безпека даних. MySQL є всесвітньо відомим тим, що є найбільш безпечною і надійною системою управління БД, яку використовують такі популярні веб-додатки: WordPress, Drupal, Joomla, Facebook та Twitter. Безпека даних і підтримка транзакційної обробки, що супроводжують останню версію MySQL, може мати велику користь для будь-якого бізнесу, особливо якщо це бізнес з електронною комерцією, що передбачає часті грошові перекази.

– Масштабованість за вимогою. MySQL пропонує неперевершену масштабованість для полегшення управління глибоко вбудованими додатками, що використовують менший розмір, навіть у великих сховищах, що складають терабайти даних. Гнучкість за вимогою є великою перевагою MySQL. Це рішення з відкритим кодом дає змогу налаштувати електронну комерцію з винятковими вимогами до сервера БД.
– Висока ефективність. MySQL має особливу систему зберігання інформації, що робить легким налаштування сервера БД MySQL для довершеної роботи. Незалежно від того, чи є це веб-сайт електронної комерції, який отримує мільйон запитів кожен день або високошвидкісну систему обробки транзакцій, MySQL розроблений для задоволення навіть найбільш вимогливих додатків, забезпечуючи оптимальну швидкість, повнотекстові індекси та унікальні кеші пам'яті для підвищення продуктивності.
– Круговий годинник. MySQL поставляється з гарантією безперебійної роботи цілодобово і пропонує широкий спектр високоякісних рішень, включаючи спеціалізовані кластерні сервери та конфігурації реплікації master / slave.
– Комплексна підтримка транзакцій. MySQL очолює список надійних механізмів транзакційних баз даних, доступних на ринку. З такими функціями, як повна атомна, послідовна, ізольована, довговічна підтримка транзакцій; підтримка транзакцій з різними версіями; і необмежене блокування на рівні рядків – це рішення для повної цілісності даних. Це гарантує миттєву ідентифікацію глухих контактів через серверну цілісність.
– Повне керування робочим процесом. При середньому часу завантаження та інсталяції менше 30 хвилин, MySQL означає зручність з першого дня. Незалежно від того, чи є ваша платформа Linux, Microsoft, Macintosh або UNIX, MySQL – це всеосяжне

рішення з функціями самоврядування, які автоматизують все, починаючи від розширення простору і конфігурації, до проектування даних і адміністрування баз даних.

– Зменшена загальна вартість володіння. Завдяки міграції поточних додатків баз даних до MySQL, підприємства користуються значною економією коштів на нових проектах. Надійність і простота управління дозволяють економити час усунення несправностей, який інакше витрачається при вирішенні проблем простою і продуктивності.
– Гнучкість відкритого джерела. Всі збої , які виникають у відкритому коді, можуть бути завершені цілодобовою підтримкою MySQL і компенсацією підприємств. Безпечна обробка і надійне програмне забезпечення MySQL поєднуються для забезпечення ефективних транзакцій для великомасштабних проектів. Це робить обслуговування, налагодження та оновлення швидкими та простими, одночасно покращуючи досвід користувачів [24].

Недоліки MySQL:

– Декілька проблем зі стабільністю. MySQL має тенденцію бути дещо менш надійною, ніж її колеги. Ці питання стабільності пов'язані з тим, як він виконує певні функції (наприклад, посилання, транзакції та аудит). Хоча база даних, безумовно, все ще може бути використана в світлі цих проблем, вони, як правило, роблять MySQL поганим вибором для певних випадків використання.
– Відносно низьке масштабування продуктивності. Хоча MySQL обладнаний практично безмежним обсягом даних, він іноді не справляється з занадто великою кількістю операцій в даний момент часу.
– Oracle є керівником розробки MySQL. Oracle їй майже не займається, прогрес, мабуть, зупинився, і лише за останні кілька років було випущено лише один великий випуск. Компанія не

приймає латки, розроблені спільнотою, і не намагається запропонувати користувачам з ними взаємодіяти. Для розробників дійсно немає можливості обговорювати СКБД з Oracle – і це проблема.

– Менша функціональність, ніж інші СКБД на ринку.
– База даних не є повністю сумісною з SQL та має тенденцію бути обмеженою в областях, включаючи зберігання даних, відмовостійкість та діагностику продуктивності (серед інших). Розробники можуть виявити цю відсутність функціональності розчаровує, особливо якщо вони звикли до більш повнофункціональної альтернативи [25].

2.2.3. *SQLite*

SQLite – реляційна СКБД, міститься в бібліотеці мови С. Вільна, серверна та функціональна, без проблем працює з веб-додатками. Спроектована у 2000 році.Мета дизайну SQLite полягала в тому, щоб дозволити програмі працювати без встановлення СКБД чи вимагати адміністратора БД.

На відміну від багатьох інших СКБД, SQLite не є механізмом бази даних клієнт-сервер. Скоріш вона вмонтована в кінцеву програму. SQLite є сумісною з ACID і за стандарту SQL реалізує більшу його частину. Проте вона використовує синтаксис SQL, що динамічно і слабо типізований, з цього випливає, що немає гарантії цілісності домену. SQLite є досить популярним вибором ПЗ вбудованої БД як для локального, так і для клієнтського зберігання в прикладних програмах, таких як, наприклад, веб-браузери. Це, мабуть, найбільш широко розгорнутий двигун БД, так як на сьогоднішній день, він використовується у більшості браузерів, операційниих системамах, та вбудованими системамх (наприклад, мобільними телефонами). SQLite має прив'язки до багатьох мов програмування [26].

Переваги SQLite:

– Нульова конфігурація. SQLite не потрібно "встановлювати" перед його використанням. Процедура "установки" не існує. Не існує серверного процесу, який потрібно запускати, зупиняти або налаштовувати. Адміністратору не потрібно створювати новий екземпляр бази даних або призначати користувачам права доступу. SQLite не використовує файлів конфігурації. Нічого не потрібно робити, щоб повідомити системі, що SQLite працює. Не потрібно виконувати жодних дій для відновлення після аварії системи або збою живлення. Немає нічого для усунення несправностей. SQLite просто працює.
– SQLite не потрібен сервер, так як більшість двигунів БД були зроблені як окремий серверний процес. За допомогою цієї БД процес, який хоче отримати доступ до БД, читає і записує безпосередньо з файлів БД на диску. Не існує проміжного серверного процесу. Будь-яка програма, яка має доступ до диска, може використовувати SQLite.
– Один файл бази даних. База даних SQLite є єдиним звичайним файлом диска, який може бути розташований будь-де в ієрархії каталогів. Якщо SQLite може читати файл диска, він може читати що-небудь у базі даних. Якщо файл диска та його каталог доступні для запису, SQLite може що-небудь змінити в базі даних. Файли баз даних можна легко скопіювати на флеш-пам'ять USB або надіслати електронною поштою для спільного використання.Інші двигуни баз даних SQL мають тенденцію зберігати дані як велику колекцію файлів. Часто ці файли знаходяться в стандартному розташуванні, до якого може отримати доступ тільки сам движок бази даних. Це робить дані більш безпечними, але також ускладнює доступ. Деякі двигуни баз даних SQL забезпечують можливість запису безпосередньо на диск і обхід всієї файлової системи. Це забезпечує

додаткову продуктивність, але витрачає значну складність налаштування та обслуговування.

– Стабільний крос-платформний файл бази даних. Файл бази даних, написаний на одній машині, може бути скопійований і використаний на іншій машині з іншою архітектурою. Більшість інших двигунів баз даних SQL вимагають скидання та відновлення бази даних при переході від однієї платформи до іншої і часто під час оновлення до нової версії програмного забезпечення.
– Відкрите джерело. Можна легко знайти сирцевий код безкоштовно онлайн.
– SQLite залишається однією з найбільш використовуваних систем БД у світі. Він сумісний практично з усіма ОС і є більш-менш промисловим стандартом [27].

Недоліки SQLite:

– SQLite використовується для обробки звернень HTTP з низьким та середнім трафіком.
– У більшості випадків розмір БД обмежений 2 Гб [28].

2.2.4 Firebase Cloud Firestore

Firebase Cloud Firestore – хмарна СУБД класу NoSQL. Дозволяє розробникам ПЗ зберігати і синхронізувати дані між декількома клієнтами. Компанія Firebase заснована у 2011 році, та куплена корпорацією Google у 2014. Firebase Cloud Firestore виведена із бета-версії 31 січня 2019 року.

Вона є наступником початкової системи баз даних Firebase, бази даних реального часу, і дозволяє вкладені документи і поля, а не перегляд дерев, наданий в базі даних реального часу. Працює в якості сервісу БД для додатків реального часу. Користувачі мають змогу користуватися API, берегти дані в хмарі та синхронізувати їх. Компанія надає можливість інтеграції з додатками Node.js , IOS, JavaScript, Android, Java та Objective-C. Можна регулювати різні параметри конфіденційності. Firebase SDK надає

максимальну безпеку для масштабування та зберігання інформації. Дані відправляються від клієнта до Firebase Cloud. При підтримці Google Cloud користувачі отримують величезний простір зберігання інформації [29].

Дані зберігаються у форматі JSON, та їх можна вкладувати до 32 рівнів. Це дозволяє зберігати інформацію без структурованості. Кожна одиниця інформації може містити декілька таких одиниць, тож користувачі мають можливість мати десятки тисяч даних, які будуть містити десятки тисяч даних. Рекомендується уникати вкладеності дочірніх вузлів.

Користуючись Firebase, з'єднання відбувається за допомогою WebSocket, а він набагато швидший за HTTP, тож вся інформація синхронізується автоматично через WebSocket. Порівняно з цим більша частина БД надають дані, тільки після моменту запиту користувача на їх отримання. Так, як дані зберігаються у хмарному сховищі, користувач отримує їх при авторизації до свого облікового запису з любого пристрою, єдине, що знадобиться – з'єднання з мережею Інтернет.

Переваги Firebase Cloud Firestore:

– Формат зберігання повністю відрізняється від формату SQL (Firebase використовує JSON), тому ви не зможете легко перенести цей файл.
– Інструменти звітності не будуть розташовані поблизу стандартного SQL.
– Відсутня можливість агрегації [30].

2.3. Вибір середовища розроблення програмного забезпечення

Для розробки було обрано Visual Studio Code – редактор коду, розроблений компанією Microsoft для таких ОС, як: Windows, Linux, MacOS. Включає в себе допомогу відлагоджувача, присутня підсвічування синтаксису, інтелектуальне завершення коду, рефакторинг коду, сніпети, та внутрішнє керування Git. Користувач може легко настроїти VS Code під свої смаки, починаючи встановленням додаткових розширень, та закінчуючи

зміною комбінацій клавіш чи теми. Сирцевий код є вільним та випускається під ліцензією MIT. Скомпільовані бінарні файли безкоштовні для будь-якого використання.

Код редактору заснований на фреймворку Electron, який досі використовується як засіб для розгортання додатків Node.js для робочого столу. Хоча він і створений на Electron, але від ПЗ Atom він використовує його невеликий компонент під назвою “Monaco”, який також використовується у Azure DevOps. До речі, VS Code визнаний самим популярним інструментом у розробці.

Є можливість відкривати більше одного каталогу, за замовчуванням підтримує цілий ряд мов програмування(JavaScript, TypeScript, CSS і HTML),але розширення для інших мов можна знайти безкоштовно у VS Code Marketplace, та завантажити у 1 клік. Якщо користувачу не потрібна деяка папка у дереві проекту, то він легко може виключити її. Користувачам дуже приваблива можливість VS Code створювати власні розширення будь-яких варіацій(від підтримки нових мов, тем, протоколів аналіз статичного коду, тощо. Також є можливість синхронізувати код між редактором та сервером без додаткового ПЗ [31].

Переваги VS Code:

– IntelliSense для мов програмування, це набір різних функцій, які допомагають користувачу у редагуванні коду. За замовчуванням IntelliSense надається для JavaScript, TypeScript, JSON, HTML, CSS, Less, Sass, та користувач має можливість легко завантажити IntelliSense для інших мов у VS Code Marketplace.
– Є вбудований термінал, це робить VS Code максимально зручним, так як не треба постійно міняти вікна для будь-якої потреби в командному рядку.
– Інтегрований контроль версій. Вбудована система контролю версій Git дозволяє, не витрачаючи часу, побачити зміни, які користувач

вніс у проект. Також доступні всі Git команди.VS Code працює з будь-яким локальним або віддаленим сховищем системи Git.

– Налагоджувач. На думку багатьох користувачів, однією з ключових особливостей редактора є підтримка налагоджувача. Вбудований налагоджувач значно допомагає прискорити компіляцію, редагування, тощо. За замовчуванням у VS Code є пітримка налагоджувача для Node.js, але для інших мов користувач може легко знайти розширення у вбудованому VS Code Marketplace.
– Деякі функції управління. VS Code надає нам функції мовного сервісу, такі як Peek Definition, Go to Definition, Find all References, та Rename Symbol. Ці функції можуть бути дуже корисні для кожного розробника.

Недоліки VS Code:

– Це не IDE і не призначений, щоб бути їм, так що, якщо користувач звик до багатофункціонального продукту від JetBrains, то він буде розчарований.
– Незначна підтримка мов, яких нема у Visual Studio Code за замовчуванням [32].

2.4. Додаткові технології та бібліотеки

2.4.1. *Firebase Authentication*

Оскільки вимоги до розроблюваного проекту вимагають реєстрації, авторизації та аутентифікації, то було прийняте рішення скористатися можливостями фреймворку Firebase Authentication. Він забезпечує бекенд-сервіс, легкий у використанні SDK та вже готові UI бібліотеки для аутентифікації користувачів у додатку. Він підтримує реєстрацію за допомогою паролю, мобільного номеру, популярних інтегрований провайдерів, таких як Google, Twitter, Facebook та інших. Firebase Authentication тісно інтегрується з іншими сервісами Firebase і використовує

стандарти OAuth 2.0 і OpenID Connect, тож може бути легко інтегрованим з бекендом розроблюваного додатку.

Firebase Auth включає також систему менеджменту користувачів. Розробник може зберігати деяку базову інформацію про зареєстрованих користувачів додатку [33].

2.4.2. *Highcharts*

Highcharts – бібліотека для створення графіків, написана на чистому JS. Вперше випущена у 2009 році, інструменти складання графіків були розроблені з урахуванням найбільш критичних потреб підприємств. Ця бібліотека, яка базується на базі SVG, мультиплатформенних графіків, яка активно розвивається з 2009 року. Він має надійну документацію, передові можливості реагування та провідну підтримку в галузі доступності. Highcharts є безкоштовним для некомерційного використання. Ліцензії потрібні для використання в комерційних додатках. Сирцевий код Highcharts можна завантажити та змінити маючи відповідну ліцензію [34].

Highcharts має багато можливостей налаштування, які прості у використанні, інтелектуально пристосовується до будь-якого пристрою та підтримує жести мультитач. Бібліотека графіків працює з будь-якою базовою БД або стеком серверів. Дані можуть бути надані в будь-якій формі, включаючи CSV, JSON або завантажені та оновлені в реальному часі. Є обгортки для більшості популярних мов, таких як .Net, PHP, Python, R і Java, а також iOS і Android, і для фреймворків, таких як Angular, Vue і React [35]. Використовуючи Highcharts графіки виходять сенсорними та оптимізованими для мобільних пристроїв. Опції є необов'язковими, так як у більшості випадків графіки виглядають і ведуть себе точно так, як вам потрібно, без будь-яких змін. Для всього іншого, проста структура параметрів дозволяє глибоке налаштування, а стиль може бути виконано за допомогою JavaScript або CSS. Highcharts також розширюється і підключається для фахівців, які шукають розширені анімації та

функціональні можливості. Дані можуть бути передані до Highcharts в будь-якій формі, навіть з іншого сайту, і функцією зворотного виклику, яка використовується для аналізу даних в масив. Дані можуть бути передані до Highcharts в будь-якій формі, навіть з іншого сайту, і функцією зворотного виклику, яка використовується для аналізу даних в масив. Якщо ввімкнути модуль експортування, користувачі можуть експортувати графік у формат PNG, JPG, PDF або SVG одним натисненням кнопки або друкувати діаграму безпосередньо з веб-сторінки [36].

2.4.3. Angular

Angular – це платформа для реалізації front-end у веб-додатках з відкритим сирцевим кодом на основі TypeScript, Розробкою займається Angular Team в Google та спільнота окремих осіб і корпорацій. Розроблюється по сьогодні з метою розширення браузерних програм на основі шаблону MVC, а також упровадження тестування та розробки, хоча перша його версія була розроблений в 2009 році як програмне забезпечення для обслуговування сервісів для зберігання JSON-даних, що вимірюють мегабайти, для створення корпоративних додатків. Фреймворк працює з HTML, що містить в собі допоміжні атрибути користувача, які нагадують директиви, і вказують на сторінку з моделлю, що представляє собою звичайні змінні JavaScript. Значення цих змінних задаються вручну або витягують зі статичних чи динамічних JSON-даних [37].

У цього фреймворку архітектура на основі компонентів, яка забезпечує якісніший код. Компоненти можна розглядати як невеликі фрагменти інтерфейсу, незалежні один від одного. Компоненти подібного характеру добре інкапсульовані, іншими словами, самодостатні. Розробники можуть повторно використовувати їх у різних частинах програми. Це особливо корисно в програмах корпоративної сфери, де різні системи сходяться, але можуть мати багато схожих елементів, таких як вікна пошуку, збирачі дат, списки сортування тощо [38].

2.5. Висновки до розділу

В цьому розділі було розглянуто основні засоби та інструменти для розробки веб-сервісів та СКБД. Отримана інформація була систематизована, та були остаточно вибрані інструменти для розробки веб-сервісу.

Для розробки системи було обрано мову JavaScript, так як з нею легко повністю інтегрувати HTML/CSS. Було обрано фреймворк Angular через свою ієрархічну структуру та архітектурних MVC шаблонів.

В якості СКБД було обрано Firebase Cloud Firestore, так як Firestore – БД, яка працює в реальному часі, та доступ до даних можно отримати з будь-якого пристрою, та в ній зручно зберігати великі масиви даних, що добре підходить до заданої задачі.

В якості середовища розробки було обрано VS Code через IntelliSense(допомога редагування коду), інтегровану консоль версій, можливість синхронізувати код між редактором та сервером без додаткового ПЗ, та через налагоджувач, який значно допомогае прискорити компіляцію.

3. РОЗРОБЛЕННЯ АРХІТЕКТУРИ ТА ПРОГРАМНИХ МОДУЛІВ

3.1. Функціональні вимоги до системи

Для користування веб-сервісом користувач спочатку повинен пройти аутентифікацію.

У користувачів, які не пройщли аутентифікацію, є можливість зареєструватися у системі, або пройти аутентифікацію, як анонімний користувач, або пройти аутентифікацію використовуючи соціальні мережі.

Зареєстровані користувачі мають доступ до такої функціональності та таких сторінок :

– сторінка завантаження файлів до облікового запису;

– головна сторінка;

– сторінка створення графіків та перегляду збережених звітів;

– завантаження готового звіту на веб-сервіс;

– зберігання звітів у своєму обліковому запису;

– генерування графіків з набору даних певного звіту;

– динамічне редагування даних на графіку;

– збереження створеного графіку на пристрій у різних форматах;

– відкрити вікно друку з поточним графіком;

– розвернути графік на весь екран.

Єдина різниця між анонімним та зареєстрованим користувачем є те, що зареєстрований користувач має можливість зберігати свої звіти у своєму обліковому записі.

3.2. Загальний опис архітектури

У розробці веб-сервісу для графічного представлення аналітичних звітів була вибрана клієнт-серверна архітектура.

Клієнт-серверна архітектура – це такий спосіб організації структури веб-додатків в якій завдання, поставлені перед системою умовно діляться між двома підсистемами: клієнтом та сервером.

До сервера звертаються безліч клієнтів та часто він зберігає дані, тож він є “центральною” частиною. З цього слідує, що сервер може існувати без клієнтів, а от клієнти без сервера не можуть. Клієнт – програма, примірників якої зазвичай "більше", ніж серверів. Наприклад, до одного веб-сервера (що зберігає сайт) може підключатися велике число веб-браузерів. Зазвичай саме з програмою-клієнтом працюють користувачі (тому її так і називають), з іншого боку клієнтом сервера може бути і робот.

У дуже навантажених додатках серверів теж може бути дуже багато (але це означає часто все одно означає, що клієнтів у рази більше

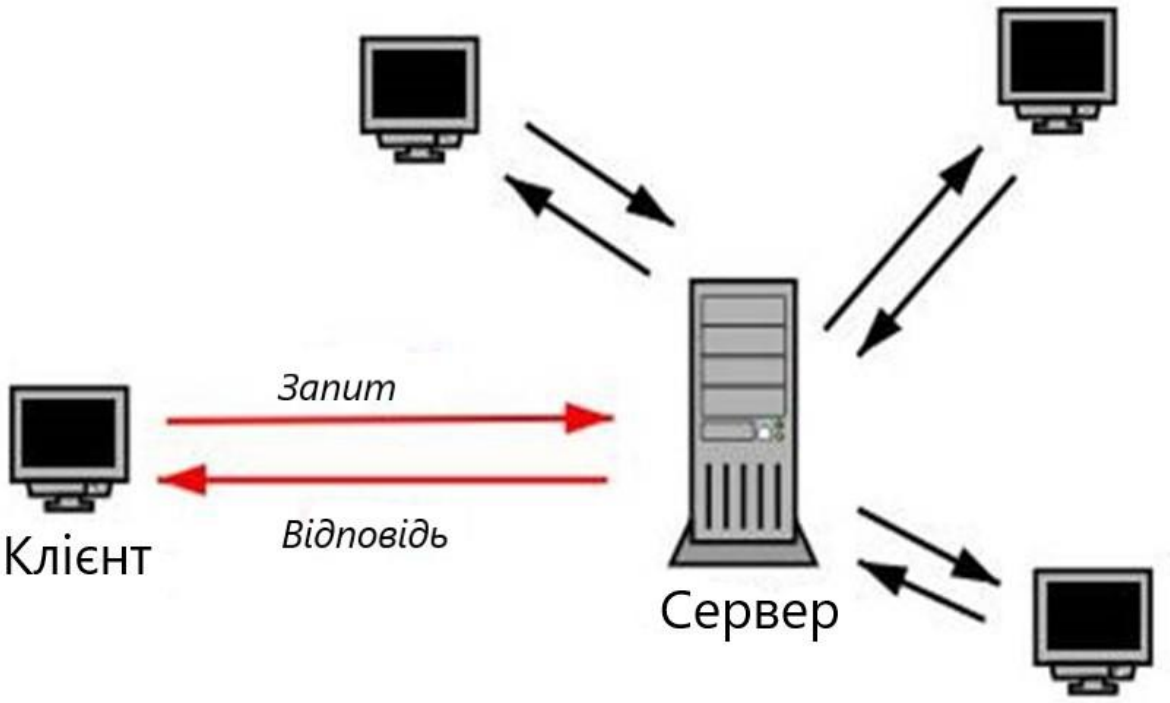


Рис. 1. Клієнт-серверна архітектура

Переваги та недоліки архітектури “клієнт-сервер”

Однією перевагою такої моделі взаємодії є те, що розділений сирцевий код клієнтського та серверного додатків. Якщо ми говоримо про локальні комп'ютерні мережі, то до переваг такої архітектури відносяться

знижені вимоги до клієнтських пристроїв, так як основна частина обчислювальних операцій буде проводитися на сервері, а також архітектура “клієнт-сервер” доволі гнучка і дає змогу адміністратору локальної мережі зробити її більш захищеною.

До недоліків такої моделі взаємодії можна віднести те, що вартість серверного обладнання в рази вище клієнтського. Сервер повинен обслуговувати спеціально навчений і підготовлений спеціаліст. Якщо в локальній мережі лягає сервер, то і клієнти не зможуть працювати (в якості окремого випадку можна привести приклад: потужності сервера не завжди вистачає, щоб задовольнить запити клієнтів, бо час очікування відповіді від сервера може бути дуже великим).

В якості висновку нам варто явно акцентувати увагу на тому, що архітектура “клієнт-сервер” не ділить машини на тільки клієнт або тільки сервер, а скоріше дозволяє розподілити навантаження і розділити функціональність між клієнтською частиною і серверної [39].

Також у розробці веб-сервісу використовувався архітектурний шаблон MVC. Він відокремлює внутрушні уявлення інформації від способів подання інформації прийняття від користувача.

Компоненти:

– Model – центральний компонент шаблону, динамічна структура даних програми, яка не залежить від інтерфейсу користувача, цей шаблон керує даними, логікою та правилами застосування.
– View – компонент, який відповідає за представлення інформації(графіки, таблиці, тощо).
– Controller – компонент, який приймає введені користувачем дані та перетворює ці дані у команди для моделі або перегляду.

Окрім розбиття програми на ці компоненти, конструкція шаблону MVC визначає взаємодію між цими компонентами.

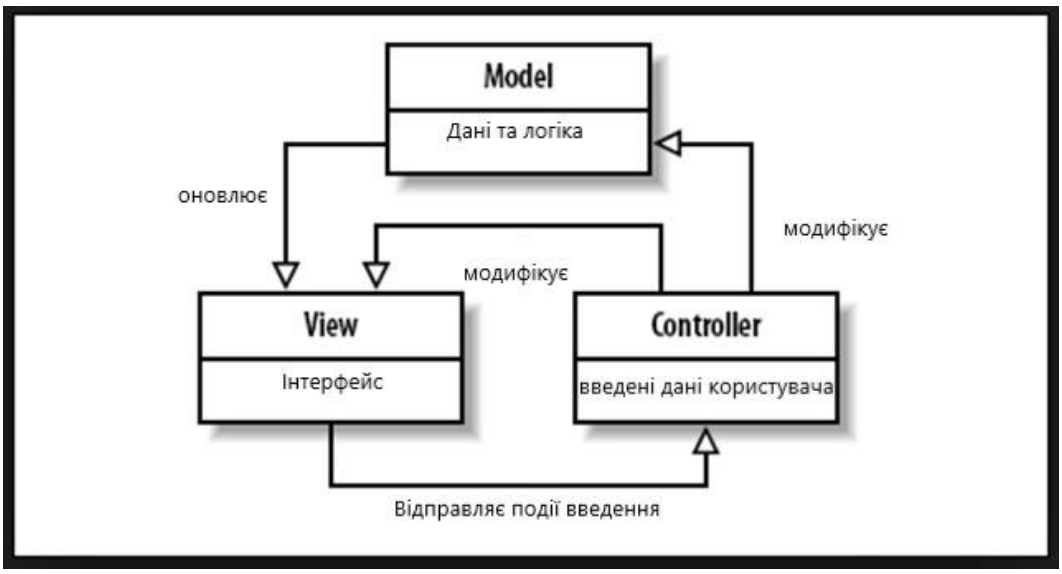


Рис. 2. Діаграма взаємодії з шаблоном MVC

Переваги використання шаблону MVC:

– Висока згуртованість. Шаблон MVC дозволяє логічне групування відповідних дій разом на контролері. View для конкретної Model також згруповані разом.
– Глобальна архітектура додатку.
– Низький зв’язок між model, view та controller.
– Легкість модифікації через поділ обов’язків.
– Спрощений механізм налагодження додатку.
– Можуть бути кілька View для однієї Model.

Недоліки використання шаблону MVC:

– Необхідність використання великої кількості ресурсу.
– Ускладнено механізм поділу програми на модулі, тому що кожен функціональний модуль повинен бути поділений на три блоки, що ускладнює архітектуру функціональних модулів програми.
– Ускладнений процес розширення функціональності.

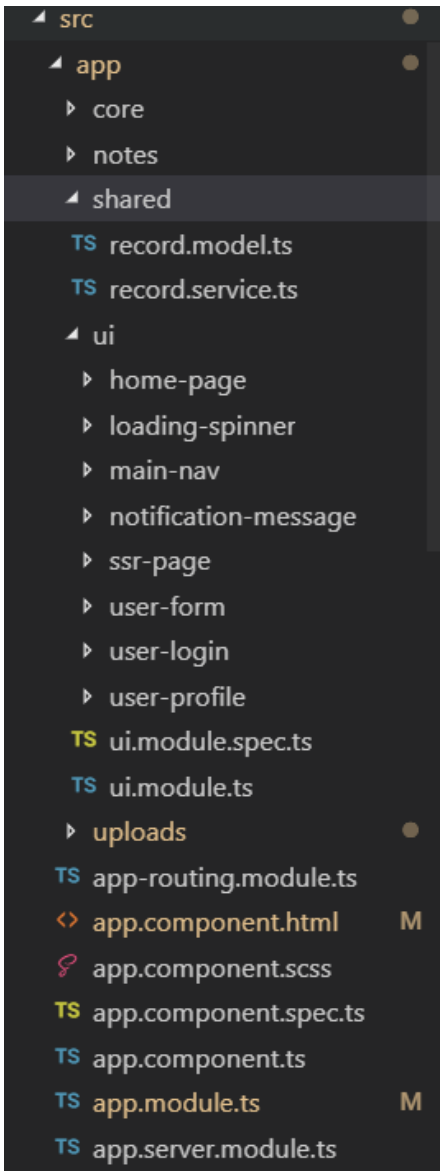


Рис. 3. Структура веб-сервісу для графічного представлення аналітичних звітів

3.3. Опис модулів програмного забезпечення

У структурі веб-сервісу можна чітко виділити такі важливі розроблені модулі:

– модуль авторизації та реєстрації;

– модуль завантаження файлів;

– модуль серверного рендерингу;

– модуль графіків;

– модуль зв’язку з БД;

– модуль зміни мови.

3.3.1. *Модуль авторизації та реєстрації*

Сутність користувача (рис. 2) складається з таких полів:

– display_name – ім’я зареєстрованого користувача, яке відображається на веб-сервісі;

– email – електронна пошта користувача, який було використано при реєстрації;

– photoURL – фотографія, яка відображається на профілі користувача;

– uid – унікальний ідентифікатор користувача;

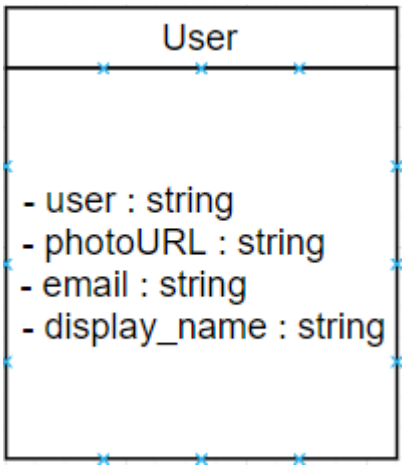


Рис. 2. Сутність користувача

Сутність користувача також пов’язана з іншою таблицею – Authentication (рис. 3).

В таблиці Authentitication зберігається наступні поля:

– creation_time – дата створення облікового запису;

– last_time – дата останнього відвідування веб-сервісу даного користувача;

– uid – унікальний ідентифікатор користувача;

– identifier – ідентифікатор користувача, наприклад, його адреса електронної скриньки.

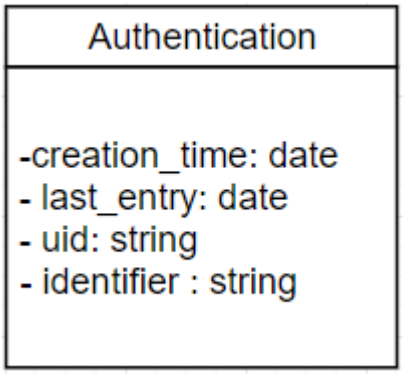


Рис. 3. Таблиця Authentication

При реєстрації користувачу треба ввести адресу електронної скриньки та пароль, також у користувача є можливість пройти аутентифікацію анонімно(якщо користувачу не буде потрібно зберігати дані, або користувач не захоче мати обліковий запис на даному веб-сервісу) Ще є можливість пройти аутентифікацію використовуючи соціальні мережі та Google. Після реєстрації з клієнта на сервер відправляються дані та пароль хешується використовуючи алгоритм “scrypt”(адаптивна криптографічна функція формування ключа на основі пароля).

3.3.2. *Модуль серверного рендерингу*

При SSR у відповідь на запит на сервері генерується усей HTML сторінки. Це виключає необхідність додаткових запитів даних з боку клієнта, так як сервер бере всю роботу на себе, перш ніж відправити відповідь. Такий підхід дозволяє домогтися швидкого рендерингу першої сторінки і рендерингу змісту першої сторінки. Швидке відображення першої сторінки може бути критично важливим для участі користувача. Приблизно 53% відвідувань сайту для мобільних пристроїв відмовляються, якщо для завантаження сторінок потрібно більше 3 секунд. Можливо, вам доведеться запускати вашу програму швидше, щоб залучити цих користувачів, перш ніж вони вирішать зробити щось інше. Виконання логіки сторінки і рендеринг на сервері дозволяють уникнути відправки клієнтові великої кількості JavaScript, що призводить до меншого часу до

інтерактивності. І це логічно, адже при серверному рендерингу користувачеві відсилаються тільки текст і посилання. Цей підхід добре спрацює на широкому діапазоні пристроїв і мережевих умов, а також відкриє можливості для цікавих браузерних оптимізацій на кшталт потокового парсинга документа.

При використанні серверного рендерингу користувачам не потрібно чекати завершення роботи JavaScript, що віднімає ресурси процесора, перш ніж вони зможуть почати працювати з сайтом. Навіть якщо не можна уникнути використання стороннього JavaScript, серверний рендеринг дозволяє зменшити кількість вашого власного JavaScript і дає більше «бюджету» для всього іншого. Однак у цього підходу є один істотний недолік: формування сторінки на сервері займає певний час, що може привести до більшого часу до першого байту [40].

У розробці використовувався Universal – модуль Angular для використання SSR (рис.4).

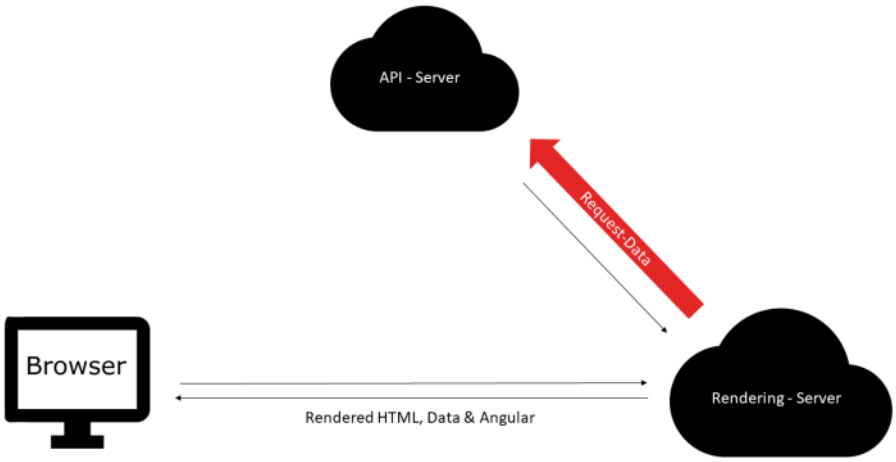


Рис. 4. Angular Universal

Звичайна програма Angular виконується в браузері, рендеринг сторінок у DOM у відповідь на дії користувача. Angular Universal виконується на сервері, генеруючи статичні сторінки додатків, які пізніше стають завантаженими на клієнта. Це означає, що програма зазвичай видає

швидше, надаючи користувачам можливість переглядати макет програми, перш ніж вона стане повністю інтерактивною.

Існує три основні причини створення Universal версії програми:

– сприяти веб-сканерам через пошукову оптимізацію (SEO);
– покращити продуктивність на мобільних та малопотужних пристроях;
– якнайшвидше показувати першу сторінку за допомогою першого вмісту (FCP).

Деякі пристрої не підтримують JavaScript або виконують JavaScript так погано, що користувацький досвід неприйнятний. У цих випадках може знадобитися версія, яка відображається на сервері, а не JavaScript. Ця версія, проте обмежена, може бути єдиною практичною альтернативою для людей, які інакше не могли використовувати програму взагалі.

За допомогою Angular Universal можна створювати цільові сторінки для програми, яка виглядає як повна програма. Сторінки є чисто HTML, і можуть відображатися, навіть якщо JavaScript вимкнено. Сторінки не обробляють події браузера, але вони підтримують навігацію по сайту за допомогою routerLink.

На практиці користувачу буде надано статичну версію цільової сторінки, щоб утримати його увагу. У той же час, сервер будете завантажувати повну програму Angular. Користувач сприймає майже миттєву продуктивність із цільової сторінки та отримує повне інтерактивне враження після повних завантажень програми.

Універсальний веб-сервер реагує на запити сторінок додатків зі статичним HTML-кодом, який видається універсальним шаблоновим механізмом. Сервер отримує і реагує на запити HTTP від клієнтів (зазвичай браузерів) і обслуговує статичні засоби, такі як скрипти, CSS і зображення. Він може відповідати на запити даних, або безпосередньо, або як проксі-сервер на окремому сервері даних.

Приклад веб-сервера для цього посібника базується на популярній рамці Express [41].

3.3.3. *Модуль графіків*

Сутність одного запису, з якого будується графік (рис. 5) складається з таких полів:

– id – унікальний ідентифікатор запису;

– allCharts – поле, в якому зберігаються всі числові дані запису;

– chart_name – назва графіку, яке йому надав користувач;

– xAxis – поле, в якому зберігаються всі назви на осі OX;

– subtitle – поле, в якому зберігається підзаголовок графіку;

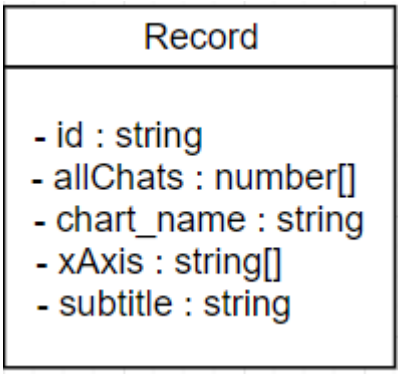


Рис. 5. Сутність запису

Використовуючи цей модуль користувач має змогу відобразити на своїй веб-сторінці графік зі свого довільного набору даних у восьми різних видах(line, spline, area, areaspline, column, bar , pie, scatter). Також завдяки цьому модулю у користувачів є можливість динамічно змінювати усі сутності запису (рис. 5) крім унікального ідентифікатора.

3.3.4. *Модуль завантаження файлів*

Користувач має можливість завантажувати свої дані на веб-сервіс використовуючи формати Excel та CSV. Спочатку файли переводяться в двійковий формат та потім проходять синтаксичний аналіз для зберігання

на хмарному сховищі як один запис (рис. 5). Є можливість дати ім’я звіту після завантаження та реалізовано завантажування декількох файлів одночасно.

3.3.5. *Модуль зв'язку з БД*

Дані зберігаються та синхронізуються з хмарною NoSQL базою даних. Дані синхронізуються між усіма клієнтами в режимі реального часу і залишаються завжди доступними. Дані зберігаються як JSON і синхронізуються в реальному часі з кожним підключеним клієнтом. Замість типових HTTP-запитів, база даних Firebase Realtime Database використовує синхронізацію даних – кожен раз, коли дані змінюються, будь-який підключений пристрій отримує це оновлення протягом декількох мілісекунд (рис. 6).

Якщо розробник будує кроссплатформовий додаток з SDK від Firebase для iOS, Android і JavaScript, то всі клієнти будуть використовувати один екземпляр бази даних Realtime і автоматично будуть отримувати оновлення з новими даними.

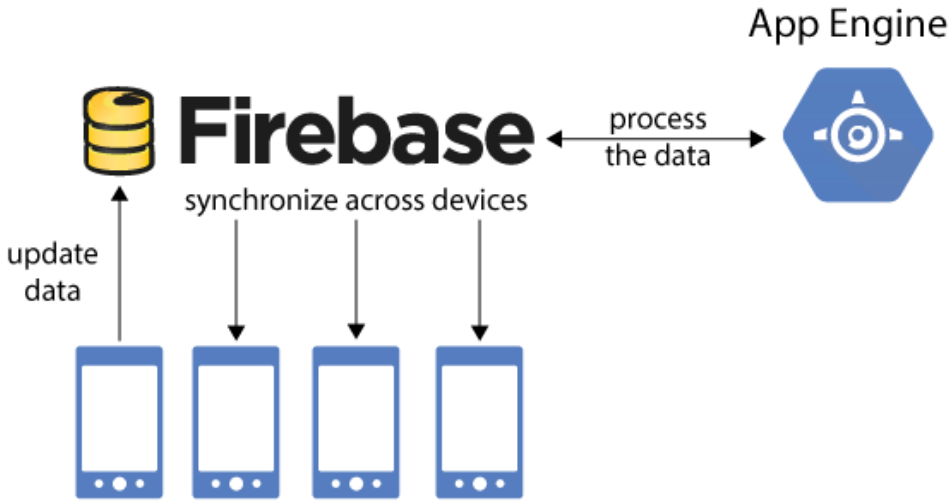


Рис. 6. Взаємодія Firebase з клієнтом

3.3.6. Модуль зміни мови

Використовується модуль i18n, який використовує динамічне сховище формату JSON, відбувається на клієнтській стороні. При переключенні мови користувачем додаток змінює використовуваний файл з мовою. У цього файлу дуже проста структура : ключ – значення, де на місті значення зберігається переклад. Завдяки такому модулю зміни мови пошукові системи знаходять контент веб-сервісів відповідно до мови запиту, і це надає великі переваги

3.4. Висновки до розділу

У цьому розділі було розглянуто основні програмні модулі розробленого веб-сервісу для графічного представлення аналітичних звітів, та було розглянуто загальний опис архітектури. Також було розглянуто взаємодію основних модулів та приведено головні сутності для роботи з цими модулями. У процесі розробки було використано наведені технології та фреймворки.

4. АНАЛІЗ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Особливості тестування веб-сервісу для графічного представлення аналітичинх звітів

У цьому підрозділі буде проведено аналіз методів тестування ПЗ.

4.1.1. *Методи тестування програмного забезпечення*

Тестування програмного забезпечення – це:

– процес дослідження ПЗ з метою отримання інформації про якість продукту;

– процес перевірки відповідності заявлених до продукту вимог і реально реалізованої функціональності, здійснюваної шляхом спостереження за його роботою в штучно створених ситуаціях і на обмеженому наборі тестів, обраних певним чином;

– оцінка системи з тим, щоб знайти відмінності між тим, якою система повинна бути і якою вона є [42].

4.1.2. *Димове тестування*

Димове тестування, також відоме як "Build Verification Testing", – це тип тестування програмного забезпечення, який складається з невичерпного набору тестів, спрямованих на забезпечення роботи найважливіших функцій. Результат цього тестування використовується, щоб вирішити, чи є розроблюване ПЗ достатньо стабільним, щоб продовжити подальше тестування.

Димове тестування охоплює більшість основних функцій програмного забезпечення, але жодну з них глибоко не перевіряє. Результат цього тесту використовується, щоб вирішити, чи слід продовжувати подальше тестування. Якщо тест на дим проходить, продовжуйте подальші випробування. Якщо це не вдасться, припиніть подальші тести і попросіть нову збірку з необхідними виправленнями. Якщо програма погано розбита, то детальне тестування може бути марною тратою часу та зусиль.

Димове тестування допомагає розкрити інтеграцію та серйозні проблеми на початку циклу. Він може проводитися як на новоствореному програмному забезпеченні, так і на розширеному програмному забезпеченні. Димове тестування виконується вручну або за допомогою засобів автоматизації / сценаріїв.

З додаванням більшої кількості функціональних можливостей і т.д., димове тестування має бути більш експансивним. Іноді в коді знадобиться лише один неправильний символ, щоб зробити цілий додаток марним.

Переваги димового тестування:

– розкриває проблеми інтеграції;
– рано відкриває проблеми;
– забезпечує певний рівень впевненості в тому, що зміни в програмному забезпеченні не вплинули негативно на основні райони (звичайно, зони, які охоплюються димовим тестуванням).

Димове тестування зазвичай використовується в рівнях інтеграційного тестування, також системою тестування та використовуючи тестування прийомів [43].

На даному етапі тестування треба перевірити функціональність описаних модулів. Робимо наступне:

– реєструємось у веб-сервісі (рис. 7);
– завантажуємо готовий звіт на веб-сервіс (рис. 8);
– відкриваємо сторінку “Records” та бачимо 1 звіт готовий для відображення, відкриваємо його (рис. 9);
– змінюємо тип графіку (рис. 10);
– вимкнемо один набір даних на графіку (рис. 11);
– динамічно змінимо деякі дані у звіті (замінимо у обведеному полі значення ‘70’ на ‘–20’) (рис. 12) та бачимо нове відображення на графіку (рис. 13);
– перевіримо завантаження зображень, зроблених зі звіту (рис. 14);
– перевіримо функцію відкривання печаті зі звітом (рис. 15).

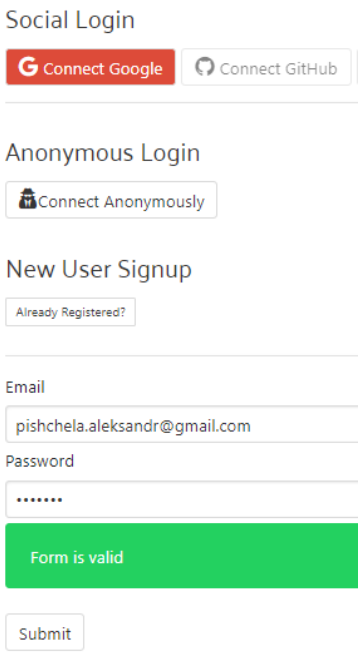


Рис. 7. Проходження реєстрації/аутентифікації

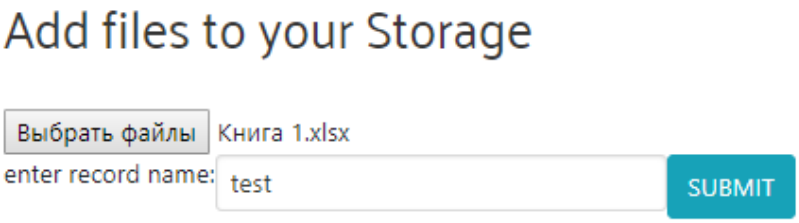


Рис. 8. Завантаження на веб-сервіс готового звіту

Рис. 9. Відображення готового звіту

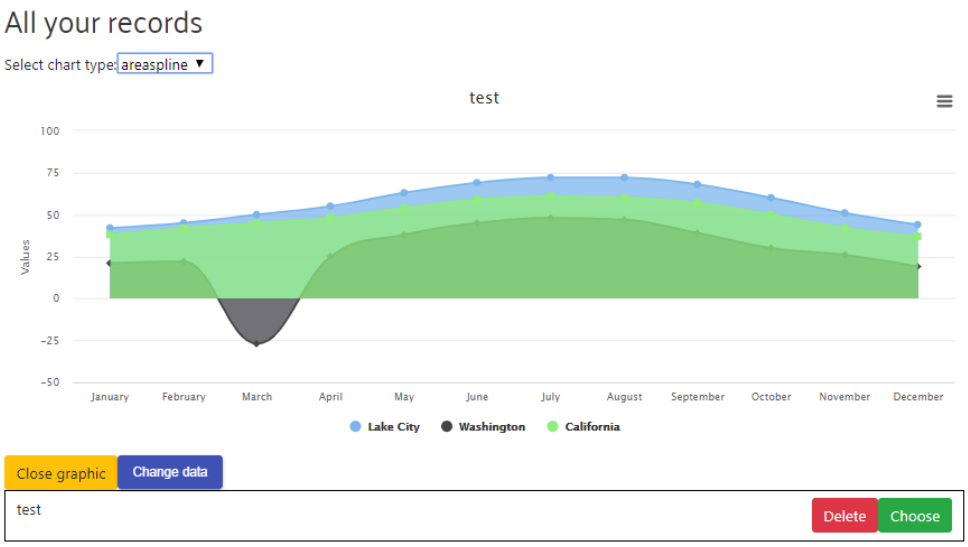


Рис. 10. Змінений тип відображення даних

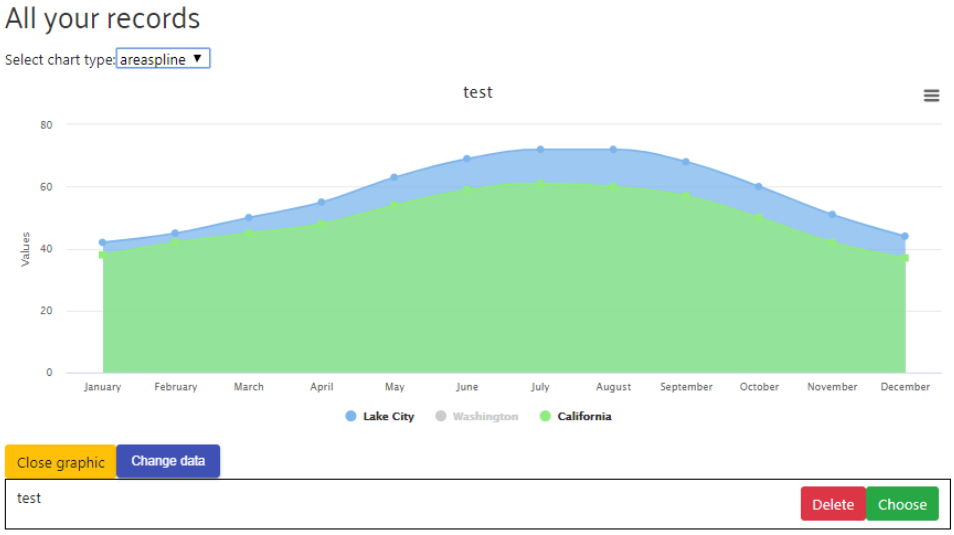


Рис. 11. Вимкнений один з наборів даних на графіку

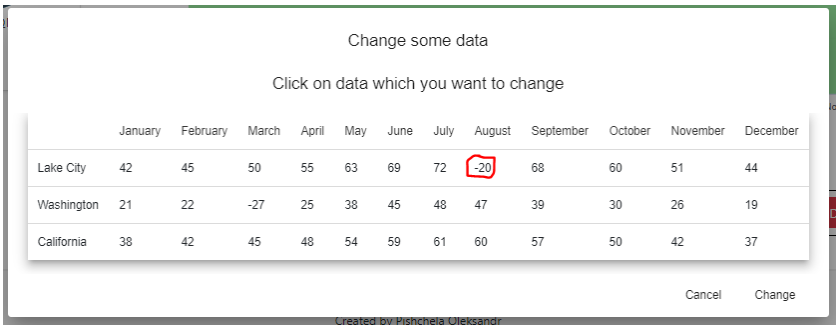


Рис. 12. Зміна деяких даних

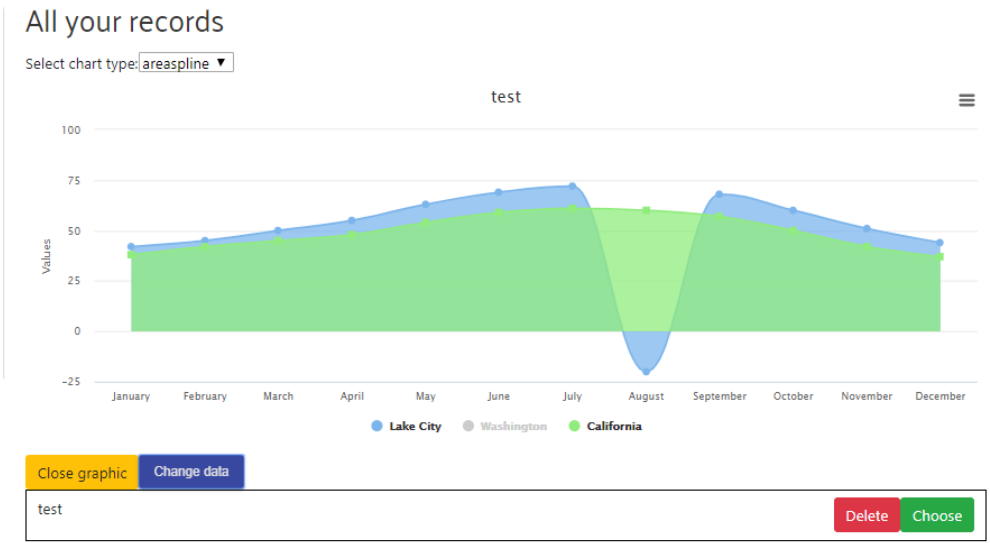


Рис. 13 Відображення зі зміненими даними

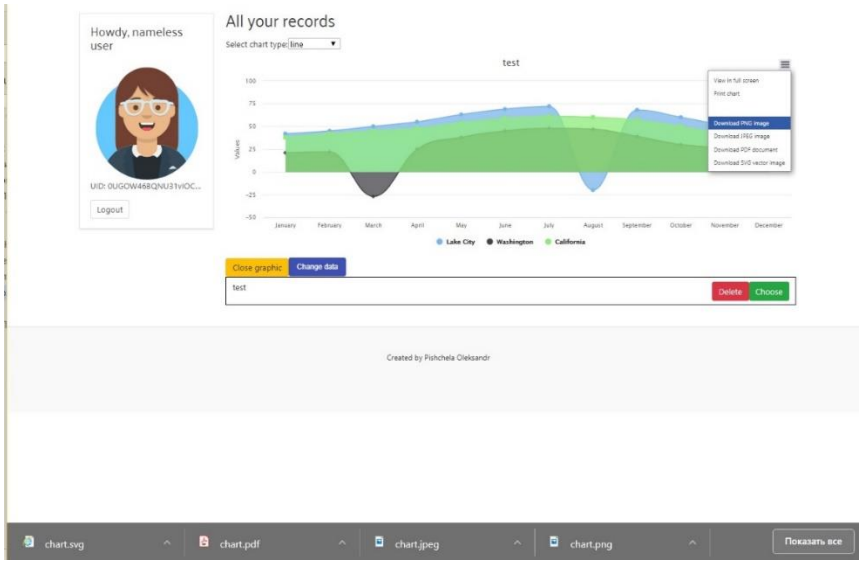


Рис. 14 Завантаження зображень з графічним відображенням звіту

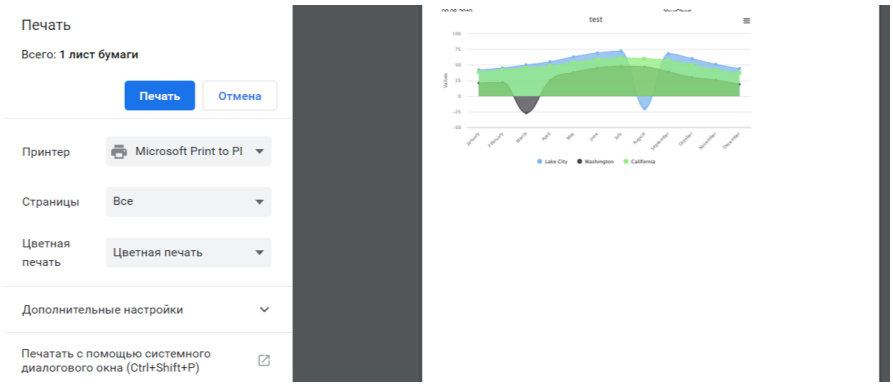


Рис. 15. Вікно печаті з графічним відображенням звіту

Димове тестування функціональності розроблених модулів ПЗ пройдене успішно.

4.2. Порівняння розробки з існуючими аналогами

Впродовж підготовки до розроблення веб-сервісу для графічного представлення аналітичних звітів було розглянуто його аналоги, їх переваги та недоліки, це детально описано в розділі 1.

Розроблений веб-сервіс має такі переваги:

– можливість розвернути графічне представлення звіту на весь екран;
– можливість динамічної зміна усіх даних звіту;
– можливість швидко переключатися між звітами;
– швидка реєстрація на веб-сервісі;
– можливість скористатися веб-сервісом анонімно(без реєстрації, коли не потрібне подальше його використання);
– можливість проходження аутентифікації використовуючи Google чи деякі соціальні мережі;
– можливість відкрити вікно для печаті з вставленим в нього графічним представленням;
– можливість збереження графічного представлення на пристрій в декількох форматах(PNG, JPEG, PDF та векторне зображення SVG).

При розробці веб-сервісу були усунуті такі недоліки існуючих аналогів, як:

– адаптованість для мобільних пристроїв;
– швидке переключення між наборами даних;
– можливість завантажити графічне представлення на пристрій;
– модуль зміни мови.

4.3. Рекомендації для подальшого вдосконалення

Подальшими напрямками удосконалення веб-сервісу для графічного представлення аналітичних звітів можна вважати наступні:

– додавання форуму, на якому користувачі будуть мати змогу ділитися деякими звітами, оцінювати їх та ставити їм відповідні оцінки;
– додавання інших типів графіків для представлення даних;
– додати модуль зміни мови, це дасть змогу використовувати сервіс більшій кількості користувачів;
– додати особистий кабінет користувача;
– зробити підтримку більшої кількості типів файлів для завантаження на сервіс звіту;
– додати можливість користувачам ділитися звітами один з одним;
– додати користувачам можливість додавання аватару;
– додати можливість будувати 3D графіки.

4.4. Висновки до розділу

В цьому розділі було проаналізовано тестування розробленого веб-сервісу для графічного представлення аналітичних звітів, також була порівняна робота розробленого ПЗ з існуючими аналогами, виявлено переваги, усунуті недоліки існуючих ПЗ. Впродовж тестування веб-сервісу помилок виявлено не було. Також були розглянуті напрямки для подальшого удосконалення веб-сервісу.

ВИСНОВКИ

Головною метою дипломного проекту було розроблення веб-сервісу для графічного представлення аналітичних звітів.

У розділі 1 були проаналізовані переваги та недоліки існуючих аналогів.

Користуючись цим аналізом в другому розділі для розробки було обрано мову JavaScript з фреймворком Angular. Також було обрано додаткові інструменти, такі як: Firebase Cloud Firestore в якості СКБД, Angular Highcharts для роботи з графіками та Firebase Authentication для реєстрації та аутентифікації. В якості середовища для розробки було обрано VS Code через його переваги перед іншими аналогами.

У третьому розділі було розглянуто основні модулі системи, основні сутності БД , та їх взаємодію між собою, також було розглянуто архітектуру розроблюваної системи. Було реалізовано функції роботи з графіками, динамічне оновлення даних, налагоджено зв’язок з БД, функції авторизації та реєстрації, завантаження файлів. Також був застосований SSR.

У четвертому розділі було зроблене тестування розроблюваного ПЗ. Також були розглянуті напрямки розвитку проекту у майбутньому, та його гнучка архітектура дає на це змогу.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Інофграфіка[Електронний ресурс]. – Режим доступу: http://catcut.net/GJmB. Дата доступу: березень 2019. Назва з екрану.
2. Поради по створенню інфографіки [Електронний ресурс]. – Режим доступу: http://catcut.net/IJmB. Дата доступу: березень 2019. Назва з екрану.
3. Топ 10 сервісів для інфографіки [Електронний ресурс]. – Режим доступу: http://catcut.net/HJmB. Дата доступу: березень 2019. Назва з екрану.
4. Історія Java [Електронний ресурс]. – Режим доступу: http://catcut.net/PJmB. Дата доступу: квітень 2019. Назва з екрану.
5. Java [Електронний ресурс]. – Режим доступу: http://catcut.net/QJmB. Дата доступу: квітень 2019. Назва з екрану.
6. Переваги Java [Електронний ресурс]. – Режим доступу: http://catcut.net/RJmB. Дата доступу: квітень 2019. Назва з екрану.
7. Недоліки Java [Електронний ресурс]. – Режим доступу: http://catcut.net/SJmB. Дата доступу: квітень 2019. Назва з екрану.
8. Python [Електронний ресурс]. – Режим доступу: http://catcut.net/KJmB .Дата доступу: квітень 2019. Назва з екрану.
9. Історія Python [Електронний ресурс]. – Режим доступу: http://catcut.net/LJmB. Дата доступу: квітень 2019. Назва з екрану.
10. Переваги Python [Електронний ресурс]. – Режим доступу: http://catcut.net/NJmB. Дата доступу: квітень 2019. Назва з екрану.
11. Недоліки Python [Електронний ресурс]. – Режим доступу: http://catcut.net/OJmB. Дата доступу: квітень 2019. Назва з екрану.
12. JavaScript [Електронний ресурс]. – Режим доступу: http://catcut.net/XJmB. Дата доступу: квітень 2019. Назва з екрану.
13. Історія JavaScript [Електронний ресурс]. – Режим доступу: http://catcut.net/YJmB. Дата доступу: квітень 2019. Назва з екрану.

14. Переваги JavaScript [Електронний ресурс]. – Режим доступу: http://catcut.net/ZJmB. Дата доступу: квітень 2019. Назва з екрану.
15. Недоліки JavaScript [Електронний ресурс]. – Режим доступу: http://catcut.net/0KmB. Дата доступу: квітень 2019. Назва з екрану.
16. PHP [Електронний ресурс]. – Режим доступу: http://catcut.net/TJmB. Дата доступу: квітень 2019. Назва з екрану.
17. Історія PHP [Електронний ресурс]. – Режим доступу: http://catcut.net/UJmB. Дата доступу: квітень 2019. Назва з екрану.
18. Переваги PHP [Електронний ресурс]. – Режим доступу: http://catcut.net/VJmB. Дата доступу: квітень 2019. Назва з екрану.
19. Недоліки PHP [Електронний ресурс]. – Режим доступу: http://catcut.net/WJmB. Дата доступу: квітень 2019. Назва з екрану.
20. PostgreSQL [Електронний ресурс]. – Режим доступу: http://catcut.net/1KmB. Дата доступу: квітень 2019. Назва з екрану.
21. Переваги PostgreSQL [Електронний ресурс]. – Режим доступу: http://catcat.net/2KmB. Дата доступу: квітень 2019. Назва з екрану.
22. Недоліки PostgreSQL [Електронний ресурс]. – Режим доступу: http://catcat.net/3KmB. Дата доступу: квітень 2019. Назва з екрану.
23. MySQL [Електронний ресурс]. – Режим доступу: http://catcut.net/4KmB. Дата доступу: квітень 2019. Назва з екрану.
24. Переваги MySQL [Електронний ресурс]. – Режим доступу: http://catcat.net/5KmB. Дата доступу: квітень 2019. Назва з екрану.
25. Недоліки MySQL [Електронний ресурс]. – Режим доступу: http://catcut.net/6KmB Дата доступу: квітень 2019. Назва з екрану.
26. SQLite [Електронний ресурс]. – Режим доступу: http://catcut.net/7KmB. Дата доступу: квітень 2019. Назва з екрану.
27. Переваги SQLite [Електронний ресурс]. – Режим доступу: http://catcat.net/8KmB. Дата доступу: квітень 2019. Назва з екрану.
28. Недоліки SQLite [Електронний ресурс]. – Режим доступу: http://catcat.net/9KmB. Дата доступу: квітень 2019. Назва з екрану.

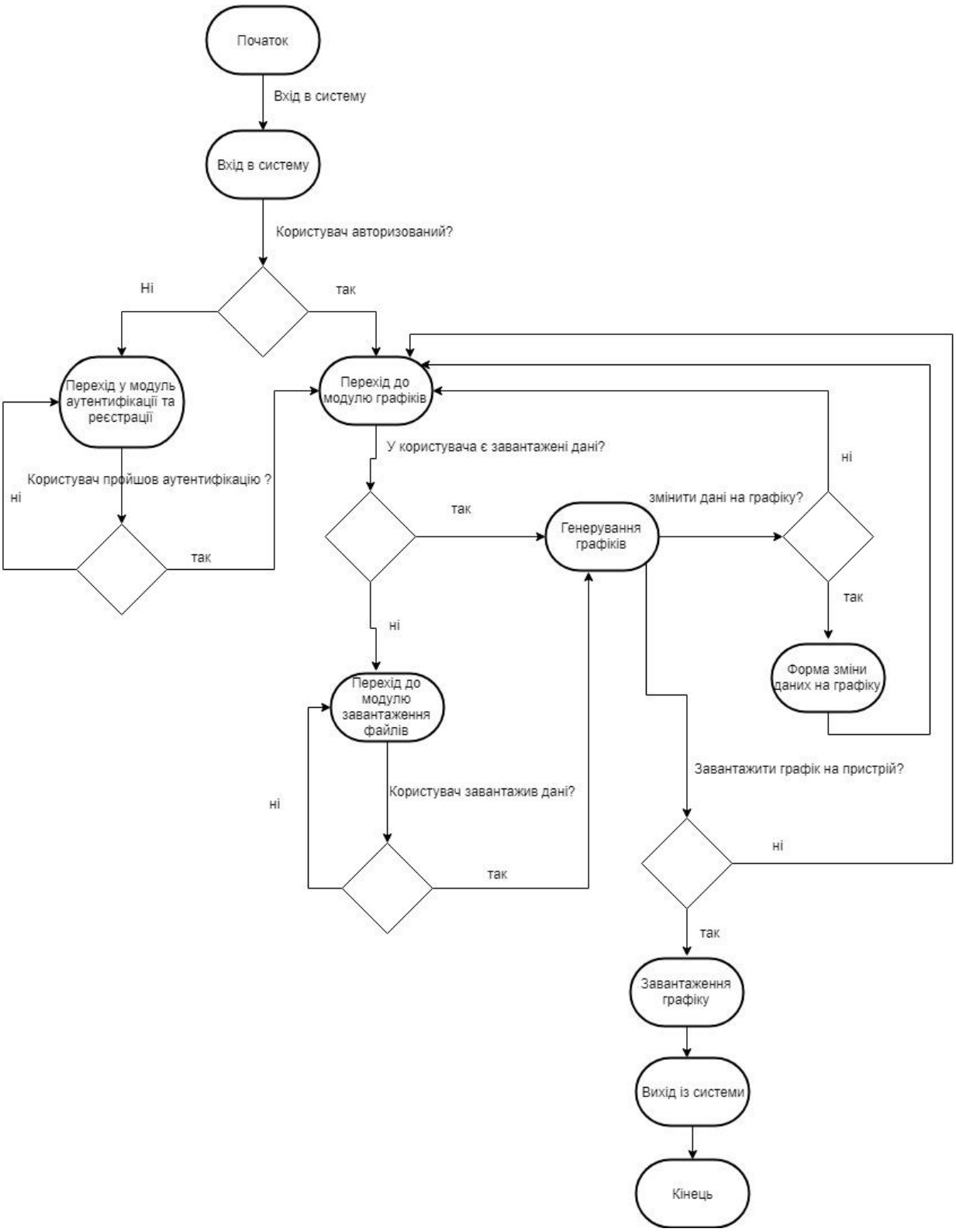
29. Firebase [Електронний ресурс]. – Режим доступу: http://catcut.net/jKmB. Дата доступу: квітень 2019. Назва з екрану.
30. Firebase Firestore [Електронний ресурс]. – Режим доступу: http://catcut.net/kKmB. Дата доступу: квітень 2019. Назва з екрану.
31. VS Code [Електронний ресурс]. – Режим доступу: http://catcut.net/iKmB. Дата доступу: квітень 2019. Назва з екрану
32. VS Code переваги та недоліки [Електронний ресурс]. – Режим доступу: http://catcut.net/hKmB. Дата доступу: квітень 2019. Назва з екрану.
33. Firebase [Електронний ресурс]. – Режим доступу: http://catcut.net/jKmB. Дата доступу: квітень 2019. Назва з екрану.
34. Highcharts [Електронний ресурс]. – Режим доступу: http://catcut.net/eKmB. Дата доступу: квітень 2019. Назва з екрану.
35. Highcharts [Електронний ресурс]. – Режим доступу: http://catcut.net/fKmB. Дата доступу: квітень 2019. Назва з екрану.
36. Highcharts [Електронний ресурс]. – Режим доступу: http://catcut.net/gKmB. Дата доступу: квітень 2019. Назва з екрану.
37. Angular [Електронний ресурс]. – Режим доступу: http://catcut.net/aKmB. Дата доступу: квітень 2019. Назва з екрану.
38. Архітектура Angular [Електронний ресурс]. – Режим доступу: http://catcut.net/bKmB. Дата доступу: квітень 2019. Назва з екрану.
39. Клієнт–серверна архітектура [Електронний ресурс]. – Режим доступу: http://catcut.net/cKmB. Дата доступу: квітень 2019. Назва з екрану.
40. Серверний рендеринг [Електронний ресурс]. – Режим доступу: http://catcut.net/dKmB. Дата доступу: квітень 2019. Назва з екрану.
41. Angular Universal[Електронний ресурс]. – Режим доступу: https://angular.io/guide/universal . Дата доступу: травень 2019.Назва з екрану.

42. Тестування програмного забезпечення[Електронний ресурс]. – Режим доступу: http://catcut.net/aynB. Дата доступу: травень 2019.Назва з екрану.
43. Димове тестування[Електронний ресурс]. – Режим доступу: http://catcut.net/iynB. Дата доступу: травень 2019.Назва з екрану.

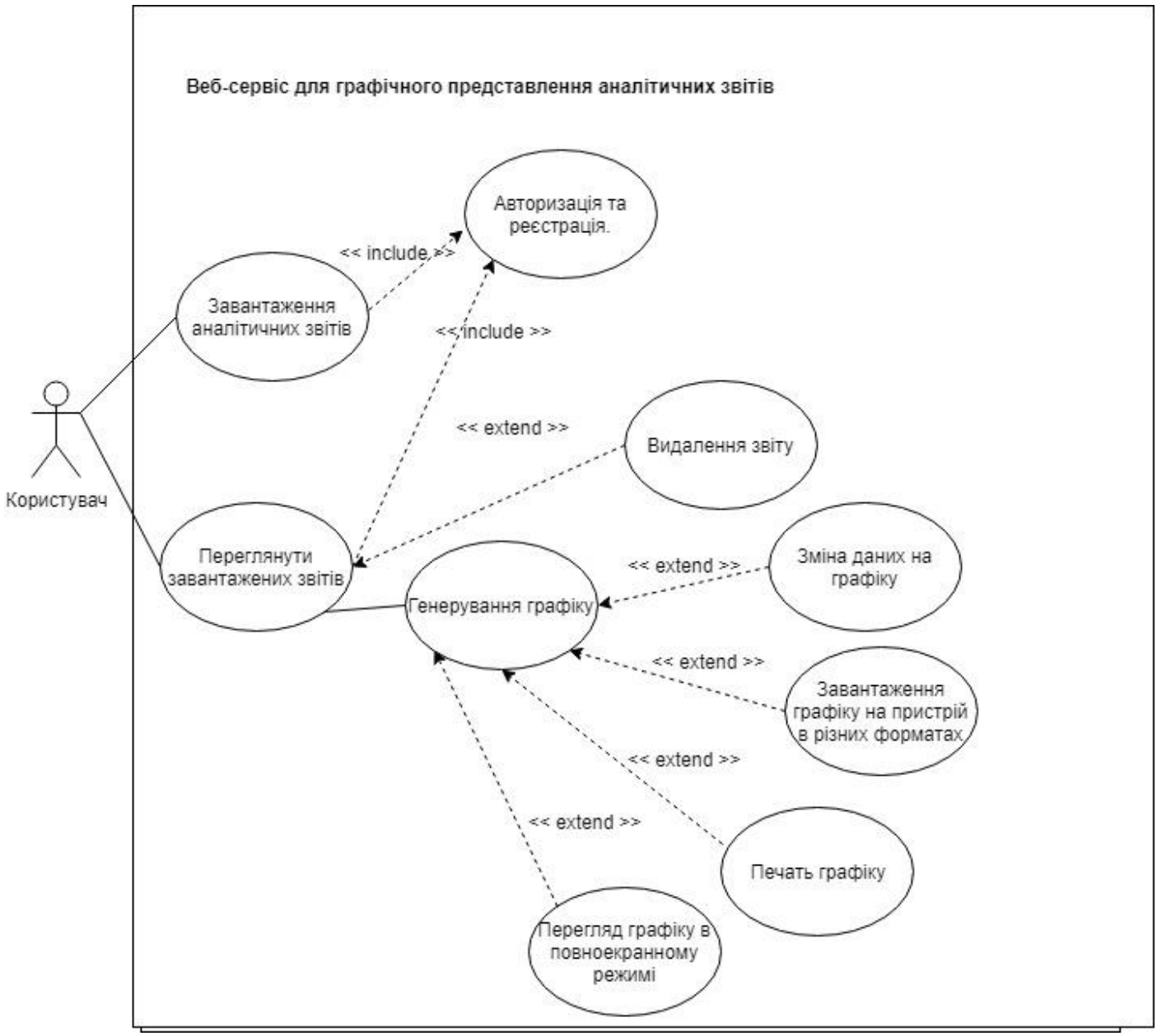
ДОДАТКИ

Додаток 1

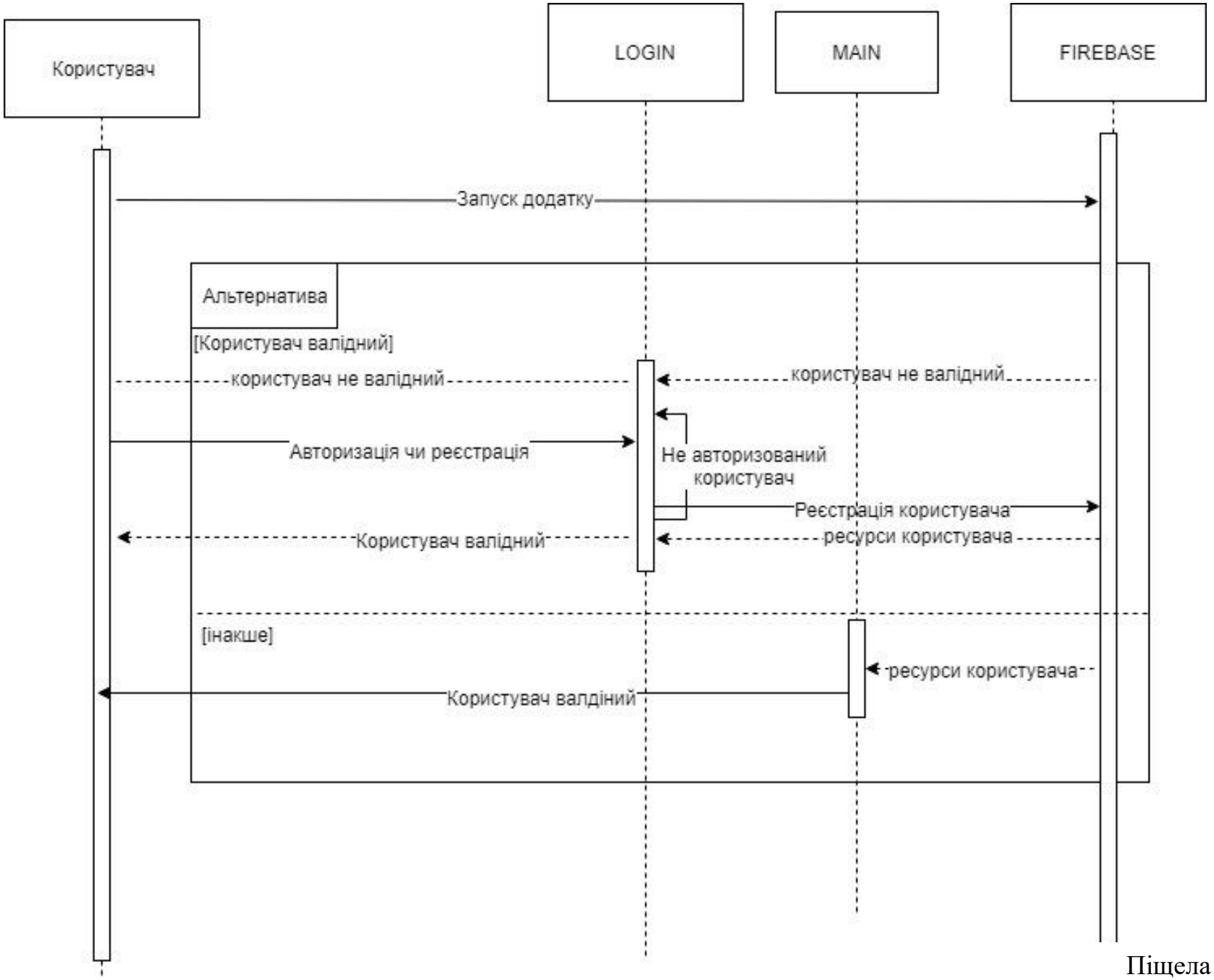
Копії графічних матеріалів



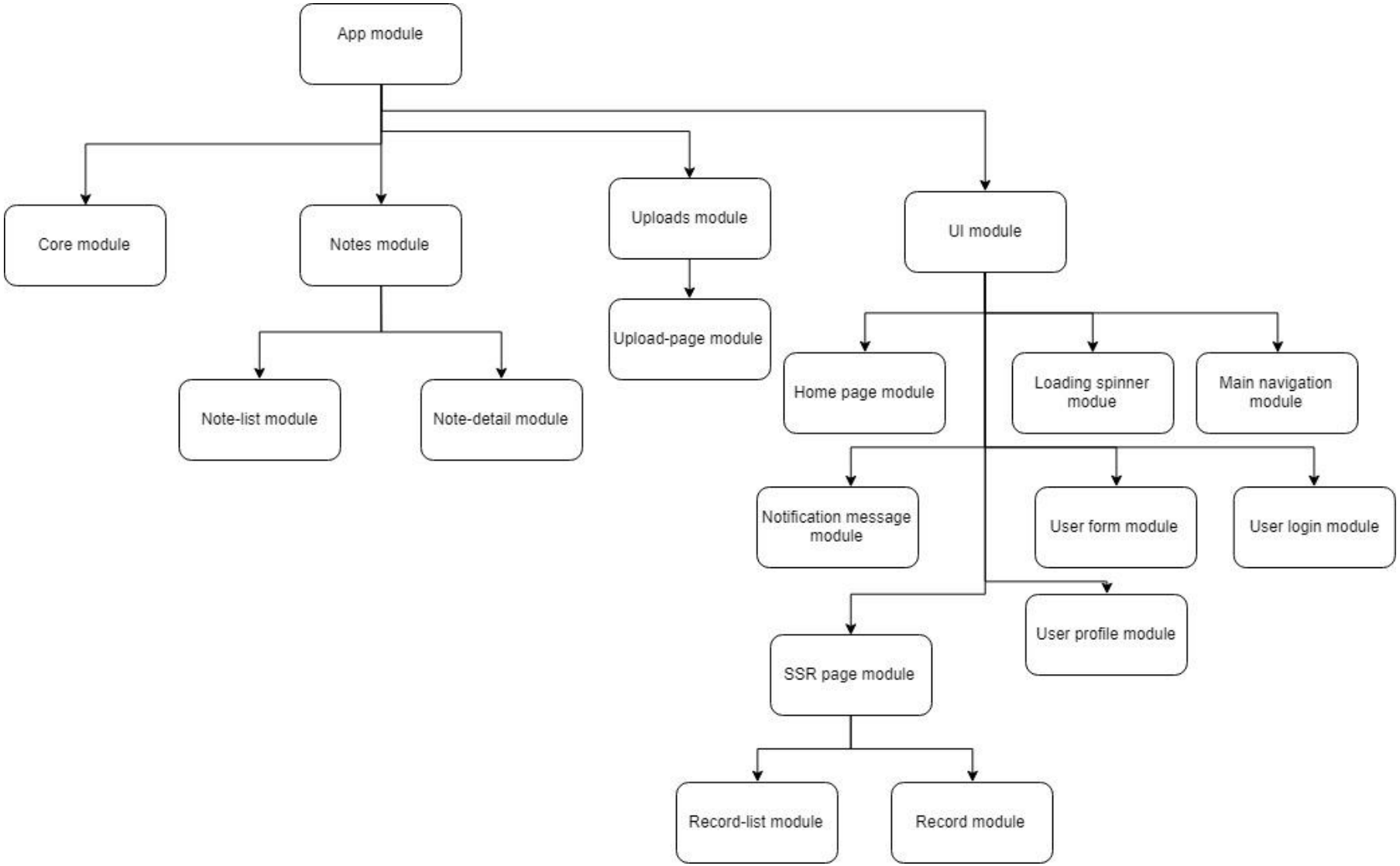
ДП.045430-07-99 Веб-сервіс для графічного представлення аналітичних звітів. Алгоритм створення графічного представлення звіту. Діаграма діяльності



Піщела О.К., група КП-52



Піщела О.К., група КП-52



ДП.045430-06-99 Веб-сервіс для графічного представлення аналітичних звітів. Зв’язок модулів. Діаграма компонентів

Додаток 2

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

ВЕБ-СЕРВІС ДЛЯ ГРАФІЧНОГО ПРЕДСТАВЛЕННЯ АНАЛІТИЧНИХ ЗВІТІВ

Виконав: Піщела Олександр Костянтинович

Науковий керівник: Старший викладач кафедри ПЗКС, к.т.н Хіцко Яна Володимірівна

Київ – 2019

ПОСТАНОВКА ЗАДАЧІ

Мета проекту: Створення веб-сервісу для графічного представлення аналітичних звітів, з допомогою якого будь-яка звичайна людина зможе створити графічне представлення потрібного їй набору даних для використання в мережі Інтернет, в презентаціях, доповідях, тощо.

ПОСТАНОВКА ЗАДАЧІ

Завдання:

1. Проаналізувати та обрати технологію для графічного подання даних.
2. Розробити додаток використовуючи обрані технології.
3. Протестувати додаток на різних наборах даних.

АКТУАЛЬНІСТЬ

- Широкий спектр застосування відображення набору даних.
- Економія часу на побудову графічного представлення даних

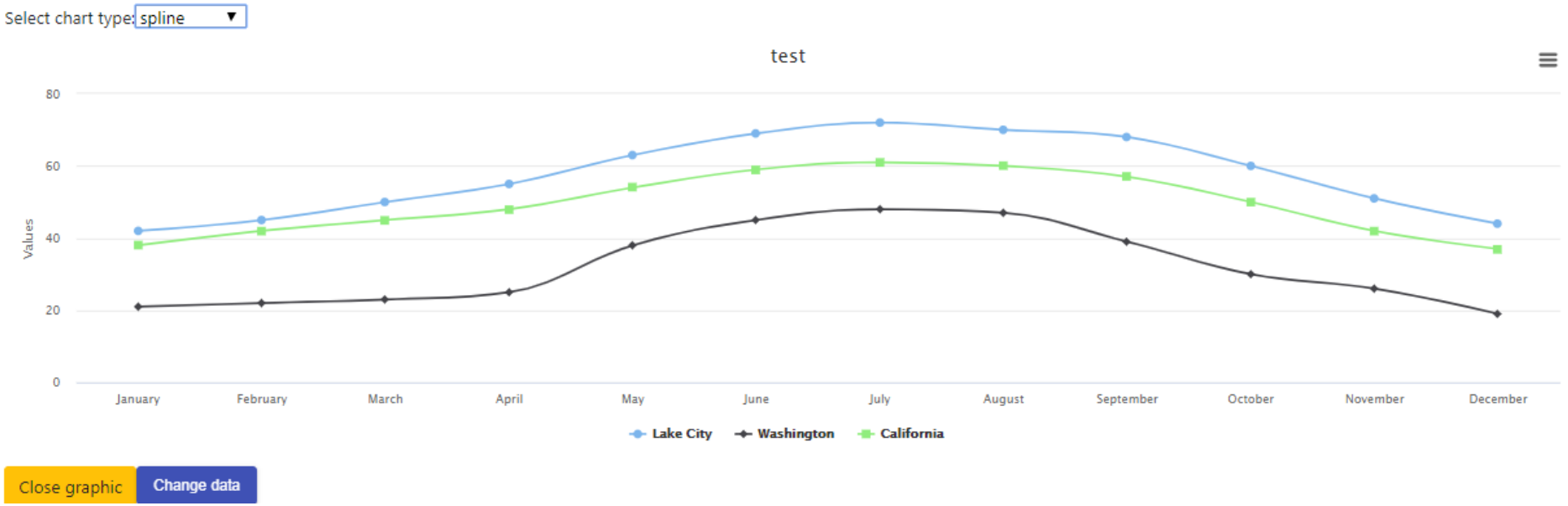
ПЕРЕВАГИ ГРАФІЧНОГО ПОДАННЯ ДАНИХ

- наочність представлення інформації;
- узагальненість поданої інформації;
- економія часу і місця для демонстрації великого обсягу даних;
- барвистість і привабливість для кінцевого користувача;

ТАБЛИЧНЕ ПОДАННЯ ДАНИХ
(таблиця зміни даних на графіку)

	January	February	March	April	May	June	July	August	September	October	November	December
Lake City	42	45	50	55	63	69	72	70	68	60	51	44
Washington	21	22	23	25	38	45	48	47	39	30	26	19
California	38	42	45	48	54	59	61	60	57	50	42	37

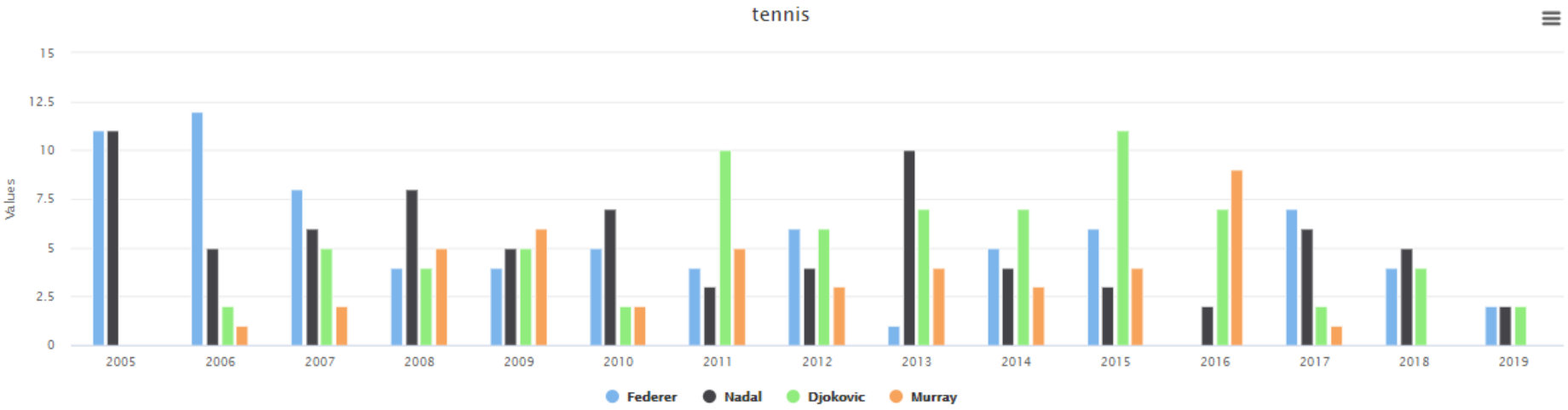
ПОДАННЯ ДАНИХ
У ВИГЛЯДІ ГРАФІКУ



ТАБЛИЧНЕ ПОДАННЯ ДАНИХ
(таблиця зміни даних на графіку)

	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
Federer	11	12	8	4	4	5	4	6	1	5	6	0	7	4	2
Nadal	11	5	6	8	5	7	3	4	10	4	3	2	6	5	2
Djokovic	0	2	5	4	5	2	10	6	7	7	11	7	2	4	2
Murray	0	1	2	5	6	2	5	3	4	3	4	9	1	0	0

ПОДАННЯ ДАНИХ
У ВИГЛЯДІ ГРАФІКУ



Аналіз існуючих рішень

Переваги та недоліки існуючих рішень

Переваги:

- багато шаблонів для представлення даних;
- Легкий UI;
- Можливість реєстрації використовуючи соціальні мережі

Недоліки:

- не адаптовані для мобільних пристроїв;
- тільки англомовні версії додатків;
- не має можливості безкоштовно завантажити створене графічне представлення даних
- невеликий обсяг сховища для одного користувача

ОБРАНІ ЗАСОБИ ВИРІШЕННЯ ПРОБЛЕМИ

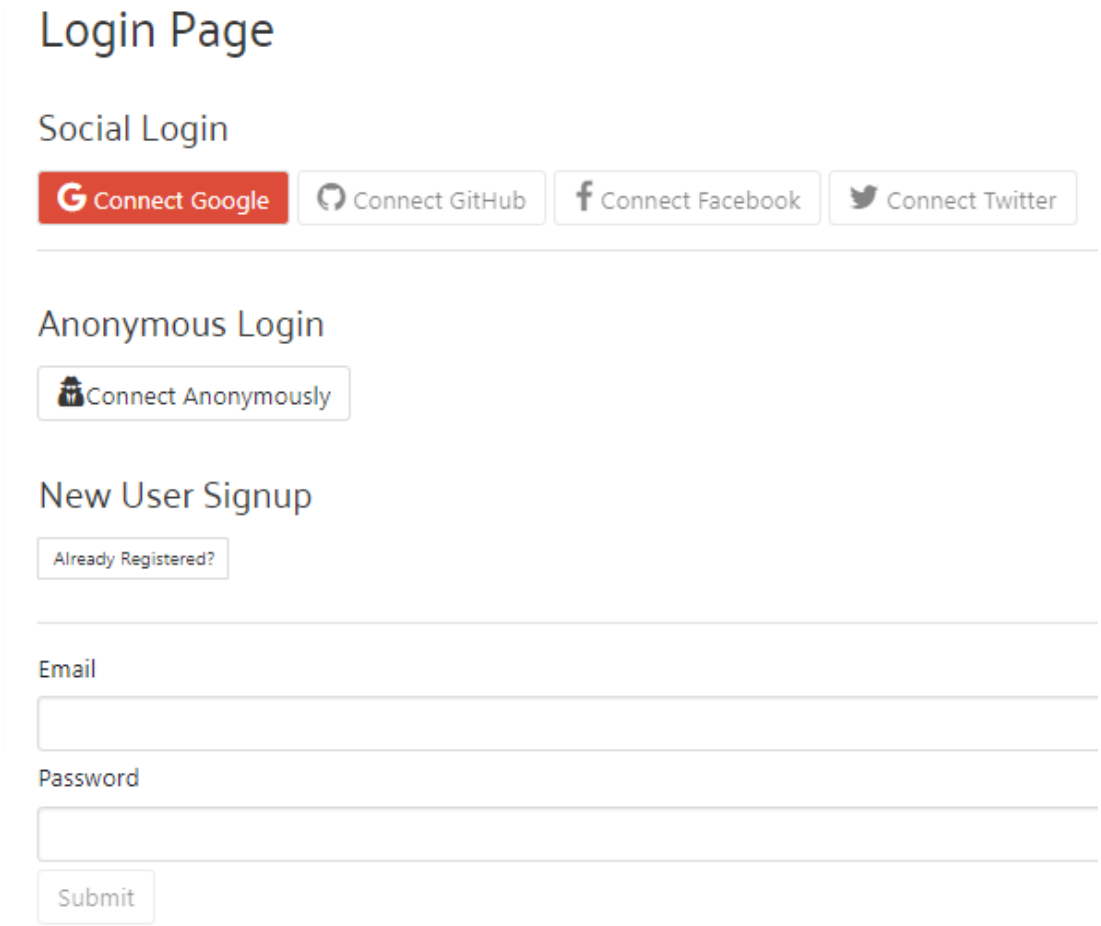
ОСНОВНІ ВИМОГИ ДО СИСТЕМИ

- Реєстрація та авторизація користувачів;
- Завантаження на сервіс звітів для подальшого використовування та зберігання;
- Генерування різних типів графіків з потрібних наборів даних;
- Можливість завантажувати згенеровані графіки на пристрій;
- Можливість динамічно оновлювати дані.

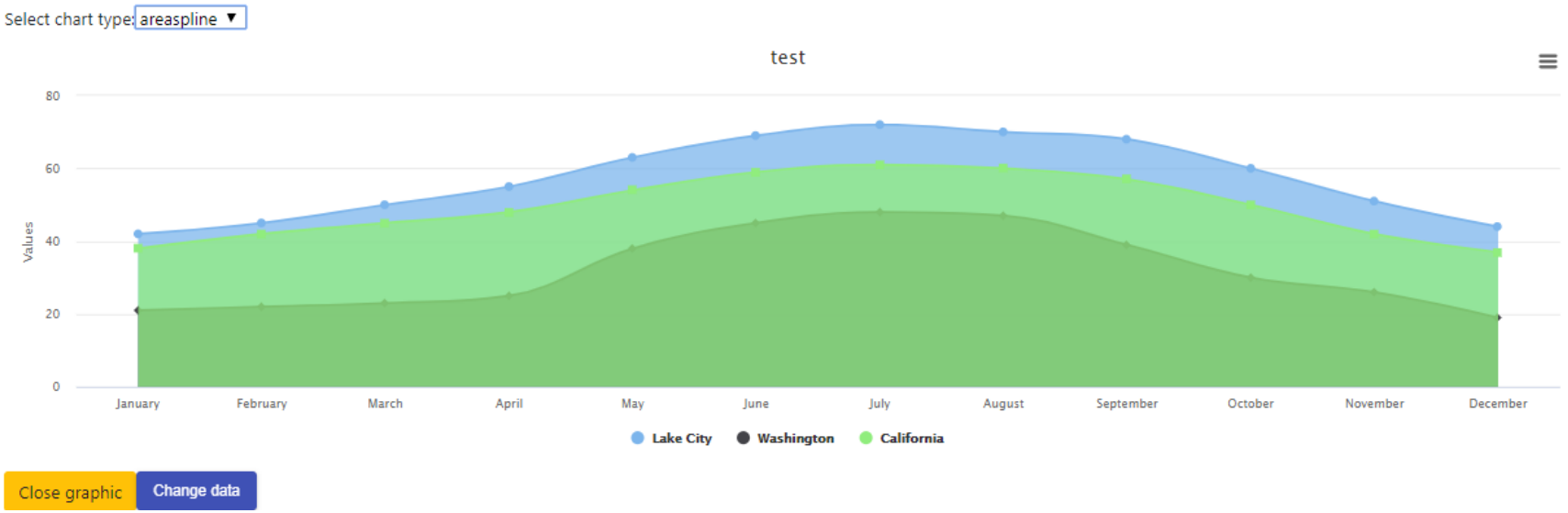
ПЕРЕЛІК ОСНОВНИХ МОДУЛІВ ВЕБ-СЕРВІСУ

- модуль авторизації та реєстрації;
- модуль завантаження файлів;
- модуль графіків;
- модуль зв’язку з БД;
- модуль зміни мови

РОЗРОБЛЕНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ (форма реєстрації/аутентифікації)



РОЗРОБЛЕНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ (відображення деякого набору даних)



РОЗРОБЛЕНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ (змінений тип відображення)

ВИСНОВКИ

- Було проведено аналіз існуючих аналогів, виділено їх переваги та недоліки;
- Користуючись цим аналізом були обрані технології для подальшої розробки;
- Розроблено веб-сервіс для графічного представлення аналітичних звітів;
- Розроблена система пройшла всі тестування успішно.

Дякую за увагу!

Додаток 3

Лістинг модулів додатку

```
import { Component, OnInit } from '@angular/core';
import {
  AngularFireStorage
} from '@angular/fire/storage';
import { AngularFirestore, AngularFirestoreDocument } from
'@angular/fire/firestore';
import { Observable } from 'rxjs';

import { Http } from '@angular/http';
import { RecordService } from 'src/app/shared/record.service';
import { Record } from 'src/app/shared/record.model';
import { AuthService } from 'src/app/core/auth.service';
import { FormGroup, FormBuilder }   from '@angular/forms';
import * as XLSX from 'xlsx';

@Component({
  selector: 'upload-page',
  templateUrl: './upload-page.component.html',
  styleUrls: ['./upload-page.component.scss']
})
export class UploadPageComponent {
///////
  form: FormGroup;
  csvData : any[] = [];
  headerRow: string[] = [];
  newRecord = new Record;
  userDoc: AngularFirestoreDocument<any>;
  user: Observable<any>;
  uid: string;
  filename : string = "";

  constructor(
    private http : Http,
    private service : RecordService,
    private firestore: AngularFirestore,
    private storage: AngularFireStorage,
    private db: AngularFirestore,
    public auth: AuthService,
    private fb: FormBuilder
  ) {
    this.form = this.fb.group({
      name: ['']
    })
  }
  onFileChange(evt: any) {
    this.filename = evt.target.files[0].name;
    const target: DataTransfer = <DataTransfer>(evt.target);
    if (target.files.length !== 1) throw new Error('Cannot use multiple
files');
    const reader: FileReader = new FileReader();
    reader.onload = (e: any) => {
      console.log(e.target);
      const bstr: string = e.target.result;
      const wb: XLSX.WorkBook = XLSX.read(bstr, {type: 'binary'});

      const wsname: string = wb.SheetNames[0];
      const ws: XLSX.WorkSheet = wb.Sheets[wsname];
      let parsedData : any[] = [];
      let result : any[] = [];
      let i = 0;
      parsedData = XLSX.utils.sheet_to_json(ws, {header: 1});
      this.headerRow = parsedData[0];
      this.newRecord.xAxis = this.headerRow.slice(1);
      parsedData.splice(0,1);
```

```
      const data = parsedData.forEach(res => {
        let dataForArray = {
          name: res[0],
          data: res.slice(1),
          type: "line"
        }
        let array: number[] = [];
        for (let i = 0; i < dataForArray.data.length;i++)
          array[i] = parseInt(dataForArray.data[i], 10);
        var obj = {
          name : dataForArray.name,
          data : array,
          type : dataForArray.type
        }
        result.push(obj);
        i++;
      }
      );
      this.newRecord.allCharts = result;
    };
    reader.readAsBinaryString(target.files[0]);
  }

  get name() {return this.form.get('name'); }
  ngOnInit() {
      this.auth.user.subscribe(user => {
        this.uid = user.uid;
    });
  }

  hanleError(err){}

  addRecord(form : FormGroup){
    this.filename = "";
    this.newRecord.chartName = form.value.name;
    this.newRecord.userUid = this.uid;
    let data = Object.assign({}, this.newRecord);
    console.log(data);
    this.firestore.collection('records').add(data);
    //}
  }
<h1>Add files to your Storage</h1>

<hr>
<div class="btn btn-primary">
  <label for="files">Choose file</label>
</div>
<p *ngIf="this.filename!=''">{{this.filename}}</p>
<input type="file" style="visibility:hidden;" id="files"
(change)="onFileChange($event)" multiple="false"/>
<form [formGroup]="form" (ngSubmit)="addRecord(form)" class="input-group">
  enter record name:<br>
  <input type="text" class="input" required [formControl]="name">
  <button
  class="btn btn-info" [disabled]="!form.valid">SUBMIT</button>
</form>

import { Component, OnInit, Inject, ViewChild } from '@angular/core';
import { RecordService } from 'src/app/shared/record.service';
import { Record } from 'src/app/shared/record.model';

import { AngularFirestore, AngularFirestoreDocument } from
'@angular/fire/firestore';
import { Chart} from 'angular-highcharts';
```

```
import 'firebase/auth';
import 'firebase/firestore';
import { AuthService } from 'src/app/core/auth.service';
import { Observable } from 'rxjs';

import {MatDialog, MatDialogConfig, MatDialogRef} from
'@angular/material/dialog';
import { RecordComponent } from '../record/record.component';
import { TitleOptions } from 'highcharts';


@Component({
  selector: 'record-list',
  templateUrl: './record-list.component.html',
  styleUrls: ['./record-list.component.scss']
})
export class RecordListComponent implements OnInit {
  list : Record[];
  types : string[] = ["line", "spline", "area", "areaspline",
  "column", "bar", "scatter"];
  currentType : string = "line";
  currentId = "1";
  chart :Chart;
  userDoc: AngularFirestoreDocument<any>;
  user: Observable<any>;
  uid: string;
  constructor(private service : RecordService,
    private firestore: AngularFirestore, public auth: AuthService, public
dialog: MatDialog) {
    }

    ngOnInit() {
      this.auth.user.subscribe(user => {
        this.uid = user.uid;
      });
      this.service.getRecords().subscribe(actionArray =>{
        this.list = actionArray.map(item => {
          return  {
            id : item.payload.doc.id,
            ...item.payload.doc.data() } as Record;
        })
      })
      this.chart = new Chart({chart: {type: this.currentType},
        title: {text: 'Linechart'},credits: {enabled: false}});
    }

openDialog() {
  let config = new MatDialogConfig();
  const dialogRef:MatDialogRef<RecordComponent> =
this.dialog.open(RecordComponent, config);
  dialogRef.componentInstance.currentId = this.currentId;
  dialogRef.componentInstance.record = this.list.filter(record => record.id
=== this.currentId)[0];
  //dialogRef.close({data : this.record});
  dialogRef.afterClosed()
  .subscribe(response => {
    if (response)
      this.changeValue(response.data);
    //console.log(response.data);
  });
}
changeValue(newRecord : Record)
{
```

```
  this.firestore.doc('records/'+ this.currentId).delete();
  let data = Object.assign({}, newRecord);
  console.log(data);
  delete data.id;
  for (let i = 0; i < data.allCharts.length; i++)
  {
    for (let j = 0; j < data.allCharts[i].data.length; j++ )
    {
      data.allCharts[i].data[j] = +data.allCharts[i].data[j];
    }
  }
  this.firestore.collection('records').add(data);
  this.onChoose(this.currentId);
}
    /*addRecord(form : FormGroup){
      this.newRecord.chartName = form.value.name;
      this.newRecord.userUid = this.uid;
      let data = Object.assign({}, this.newRecord);
      console.log(data);
      this.firestore.collection('records').add(data);
      //}
    }*/
    onChoose(id : string)
    {
      for (let i = 0; i < this.list.length; i++)
      {
        if (this.list[i].id == id)
        {
          if (this.currentId != "1")
            this.onClose();
          this.currentId = id;
          for (let j = 0;j < this.list[i].allCharts.length; j++)
            this.chart.addSeries({name : this.list[i].allCharts[j].name,
              data : this.list[i].allCharts[j].data, type :
this.list[i].allCharts[j].type }, true, false);
            this.chart.ref$.subscribe(ref => {
              this.chart.ref.setTitle({text : this.list[i].chartName},{text
: ""}, true);
              this.chart.ref.xAxis[0].setCategories(this.list[i].xAxis);
            });


          }
      }
    }

    ChangeCharType(ChosenType : string){
      if (ChosenType != this.currentType)
      {
        this.currentType = ChosenType;
        while (this.chart.ref.series.length > 0)
          this.chart.removeSeries(0);
        for (let i = 0;i < this.list.length; i++)
        {
          if (this.list[i].id == this.currentId)
          {
            for (let j = 0;j < this.list[i].allCharts.length; j++)
            {
              this.list[i].allCharts[j].type = ChosenType;
              this.chart.addSeries({name : this.list[i].allCharts[j].name,
                data : this.list[i].allCharts[j].data, type :
this.list[i].allCharts[j].type }, true, false);
            }
            //this.chart.ref.xAxis[0].setCategories(this.list[i].xAxis);
```

```
          }
        }
      }
    }

    onClose()
    {
      while (this.chart.ref.series.length > 0)
          this.chart.removeSeries(0);
      this.currentId = "1";
    }
    onDelete(id :string){
      if (confirm("Are you sure to delete this record?"))
      {
        this.firestore.doc('records/'+ id).delete();
      }
    }
}
<div *ngIf="this.currentId != '1'">
    <label>Select chart type: </label>
    <select (change)="ChangeCharType($event.target.value)">
        <option *ngFor="let type of types" value={{type}}>
            {{type}}
        </option>
    </select>
  </div>

<div *ngIf="this.currentId != '1'" [chart]="chart"></div>
<button *ngIf="this.currentId != '1'" class="btn btn-warning"
(click)="onClose()">Close graphic</button>
<button *ngIf="this.currentId != '1'" mat-raised-button color="primary"
(click)="openDialog()">Change data</button>
<table class="table table-hover">
  <tbody>
    <tr class="da" *ngFor="let record of list">
      <ng-container *ngIf="record.userUid == uid">
      <!-- <div class="my-div" *ngIf="record.userUid == uid"> -->
        <td class="my-names">{{record.chartName}}</td>
        <td  class="my-buttons">
          <button class="btn btn-danger"
(click)="onDelete(record.id)">Delete</button>
          <button class="btn btn-success"
(click)="onChoose(record.id)">Choose</button>
        </td>
      </ng-container>
      <!-- </div> -->
    </tr>
  </tbody>
</table>

import { Component, OnInit, Inject, } from '@angular/core';
import { RecordService } from 'src/app/shared/record.service';
import { AngularFirestore } from '@angular/fire/firestore';
import { Record } from 'src/app/shared/record.model';
import { MAT_DIALOG_DATA, MatDialogRef } from '@angular/material';
import { AuthService } from 'src/app/core/auth.service';

//import { RecordListComponent } from '../record-list/record-
list.component';

@Component({
  selector: 'record',
  templateUrl: './record.component.html',
  styleUrls: ['./record.component.scss']
```

```
})
export class RecordComponent implements OnInit{
  list : Record[];
  uid: string;
  currentId : string;
  tableHeader: string[];
  record: Record;
  isClicked : boolean = false;

  currentType : string = null;
  currentValue : string = null;
  curFirst : number;
  curSecond : number;
  constructor(private service : RecordService,
    private firestore: AngularFirestore,
    public auth: AuthService, public dialogRef:
MatDialogRef<RecordComponent>) {
     }
     onClick(i : string, a : string, k: number)
     {
       this.curFirst = k;
       this.isClicked = true;
       this.currentType = a;
       this.currentValue = i;
     }

     onClickD(i : string, a : string, j: number, z: number)
     {
       this.isClicked = true;
       this.curFirst = j;
       this.curSecond = z;
       this.currentType = a;
       this.currentValue = i;
     }

     onKey(event :any){
        if (this.currentType == 'name')
          this.record.allCharts[this.curFirst].name = event.target.value;
        else if (this.currentType == 'Xaxis')
          this.record.xAxis[this.curFirst] = event.target.value;
        else if (this.currentType == 'data')
          this.record.allCharts[this.curFirst].data[this.curSecond] =
+event.target.value;
        this.isClicked = false;
     }

     theEnd(){
       console.log(this.record);
        this.dialogRef.close({data : this.record});
      //this.dialogRef.close(`${this.record}`);
      }

<h2 mat-dialog-title class="h1text">Change some data</h2>
<h5 mat-dialog-title class="h1text">Click on data which you want to
change</h5>
<mat-dialog-content class="mat-typography">
    <div class="mat-elevation-z8">
        <div class="table-responsive-xl">
      <table class="table">
        <thead>
          <tr>
            <td></td>
            <td *ngFor="let i of record.xAxis;let k = index"
(click)="onClick(i, 'Xaxis', k)">{{i}}</td>
```

```
          </tr>
        </thead>
        <tbody>
            <tr *ngFor="let i of record.allCharts;let j = index">
              <td (click)="onClick(i.name, 'name', j)">{{i.name}}</td>
              <td *ngFor="let a of i.data;let z = index"
(click)="onClickD(a, 'data', j, z)">{{a}}</td>
            </tr>
        </tbody>
      </table>
      </div>
    </div>
</mat-dialog-content>
<mat-dialog-actions align="end">
  <input matInput *ngIf="isClicked == true" value="{{currentValue}}"
(keyup.enter)="onKey($event)">
  <button mat-button mat-dialog-close>Cancel</button>
  <button mat-button [mat-dialog-close]="true" cdkFocusInitial
(click)="theEnd()">Change</button>
</mat-dialog-actions>

<div id="wrapper" class="content">
    <main-nav></main-nav>

    <div class="columns">
        <aside class="column is-2 is-offset-1">
            <user-profile></user-profile>
        </aside>

        <main class="column is-8">
            <notification-message></notification-message>

            <router-outlet></router-outlet>
        </main>
    </div>

    <footer class="footer" i18n="Main Footer | footer with
name@@mainfooter">
        Created by Pishchela Oleksandr
    </footer>

    <div class="github-banner" style="z-index: 9999999">
        <a href="https://github.com/pishchela" class="github-corner" aria-
label="View source on Github">
            <svg width="60" height="60" viewBox="0 0 250 250" style="z-
index: 11; fill:#FD6C6C; color:#fff; position: absolute; top: 0; border: 0;
right: 0;" aria-hidden="true"><path d="M0,0 L115,115 L130,115 L142,142
L250,250 L250,0 Z"></path><path d="M128.3,109.0 C113.8,99.7 119.0,89.6
119.0,89.6 C122.0,82.7 120.5,78.6 120.5,78.6 C119.2,72.0 123.4,76.3
123.4,76.3 C127.3,80.9 125.5,87.3 125.5,87.3 C122.9,97.6 130.6,101.9
134.4,103.2" fill="currentColor" style="transform-origin: 130px 106px;"
class="octo-arm"></path><path d="M115.0,115.0 C114.9,115.1 118.7,116.5
119.8,115.4 L133.7,101.6 C136.9,99.2 139.9,98.4 142.2,98.6 C133.8,88.0
127.5,74.4 143.8,58.0 C148.5,53.4 154.0,51.2 159.7,51.0 C160.3,49.4
163.2,43.6 171.4,40.1 C171.4,40.1 176.1,42.5 178.8,56.2 C183.1,58.6
187.2,61.8 190.9,65.4 C194.5,69.0 197.7,73.2 200.1,77.6 C213.8,80.2
216.3,84.9 216.3,84.9 C212.7,93.1 206.9,96.0 205.4,96.6 C205.1,102.4
203.0,107.8 198.3,112.5 C181.9,128.9 168.3,122.5 157.7,114.1 C157.9,116.9
156.7,120.9 152.7,124.9 L141.0,136.5 C139.8,137.7 141.6,141.9 141.8,141.8
Z" fill="currentColor" class="octo-body"></path>
      </svg>
        </a>
        <style>
            .github-corner:hover .octo-arm {
```

```
            animation: octocat-wave 560ms ease-in-out
        }

        @keyframes octocat-wave {
            0%,
            100% {
                transform: rotate(0)
            }
            20%,
            60% {
                transform: rotate(-25deg)
            }
            40%,
            80% {
                transform: rotate(10deg)
            }
        }

        @media (max-width:500px) {
            .github-corner:hover .octo-arm {
                animation: none
            }
            .github-corner .octo-arm {
                animation: octocat-wave 560ms ease-in-out
            }
        }
    </style>
</div>
</div>
```

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

ВЕБ-СЕРВІС ДЛЯ ГРАФІЧНОГО ПРЕДСТАВЛЕННЯ АНАЛІТИЧНИХ ЗВІТІВ

Програма та методика тестування

ДП.045440-04-51

"ПОГОДЖЕНО"

Керівник проекту:

_____ Я.В. Хіцко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ О.К. Піщела

2018

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Веб-сервіс для графічного представлення аналітичних звітів являє собою web-сайт, створений з використанням технологій Angular, highcharts, Firebase, та серверного рендерингу.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1) коректність роботи з базою даних;
2) перевірка аутентифікації;
3) тестування верстки;
4) функціональне тестування;
5) перевірка завантаження файлів;
6) тестування кросбраузеронсті;
7) usability тестування;
8) тестування продуктивності сайту;
9) відповідність вимогам технічного завдання;

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом “Трьох Ящиків”. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам.

Використовуються наступні методи:

1) Перевірка “чорного ящику”.
2) Перевірка “білого ящику”.
3) Перевірка “сірого ящику”.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується використовуючи засоби інструментарію qTest. Працездатність програмного забезпечення перевіряється шляхом:

1) динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати;
2) динамічного ручного тестування на відповідність функціональним вимогам;
3) статичного тестування коду;
4) тестування кросбраузерності в різних web-браузерах;
5) тестування верстки;
6) тестування зручності використання;
7) тестування інтерфейсу;
8) тестування стабільності роботи при різних умовах;

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ВЕБ-СЕРВІС ДЛЯ ГРАФІЧНОГО ПРЕДСТАВЛЕННЯ АНАЛІТИЧНИХ ЗВІТІВ

Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проекту:

_____ Я.В. Хіцко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ О.К. Піщела

2019

ЗМІСТ

1. Опис структури програмного забезпечення

Програмне забезпечення веб-сервісу для графічного представлення аналітичних звітів складається з набору модулів. Основне призначення – створення графіків з наборів даних. Структура програмного забезпечення складається з таких модулів:

– модуль авторизації та реєстрації;

– модуль завантаження файлів;

– модуль серверного рендерингу;

– модуль графіків;

– модуль зміни мови;

– модуль зв’язку з БД.

2. Опис роботи з модулями програмного забезпечення

Модуль реєстрації (рис. 1) складається з набору полів, які користувачу необхідно заповнити:

– поле електронної адреси;

– поле паролю;

Якщо користувач вводить не правильну інформацію, то поле буде виділено червоним, і з’явиться повідомлення про помилку. Після того як користувач заповнив всі поля він натискає на кнопку реєстрації та може користуватися веб-сервісом. Також у користувача є можливість зайти на веб-сервіс як анонімний користувач або використовуючи Google.

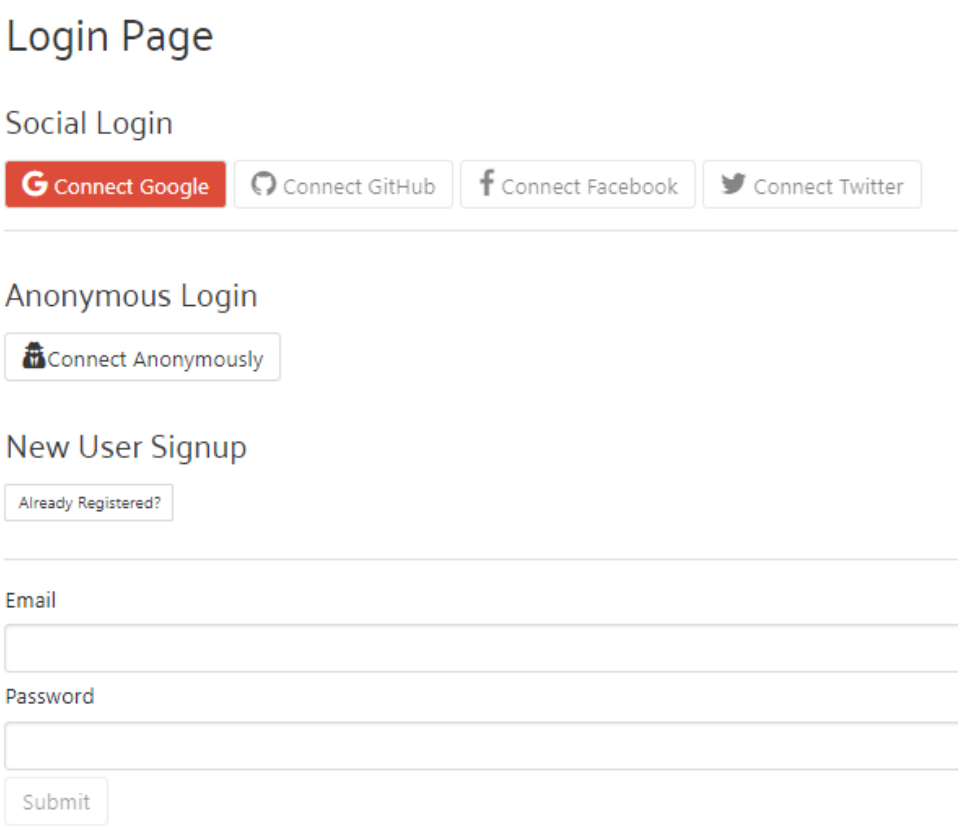


Рис. 1. Модуль реєстрації

Тепер користувачу треба завантажити свій звіт з набором даних для подальшої роботи з графіками (рис. 2).

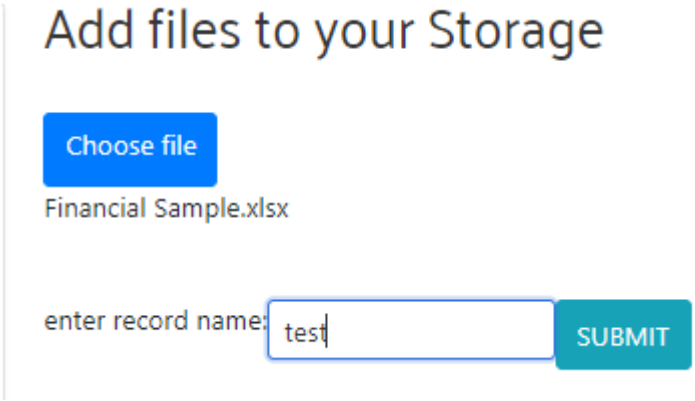


Рис. 2. Модуль завантаження файлів

Після завантаження файлу із набором даних користувач може перейти на сторінку Records в якій реалізовано модуль графіків, де будуть відображені всі завантажені ним звіти (рис. 3)

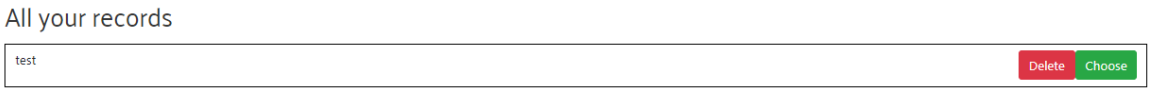


Рис. 3. Список завантажених звітів користувачем

Натиснувши на клавішу “Choose” згенерується графічне представлення звіту (рис. 4)

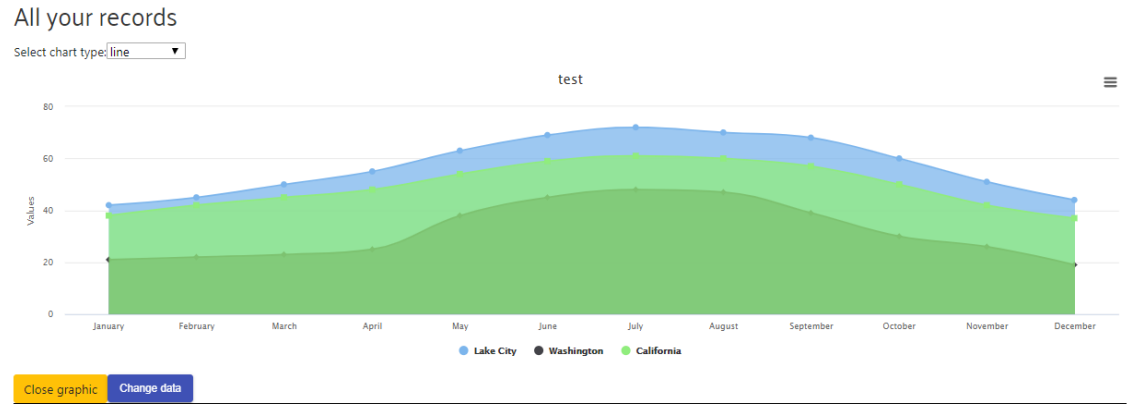


Рис. 4. Згенерований звіт

Тепер користувач має змогу редагувати звіт ,змінимо деякі дані у аналітичному звіті (у обведеному полі значення ‘70’ на ‘–20’ (рис. 5), та дані будуть динамічно оновлені на графіку (рис. 6).

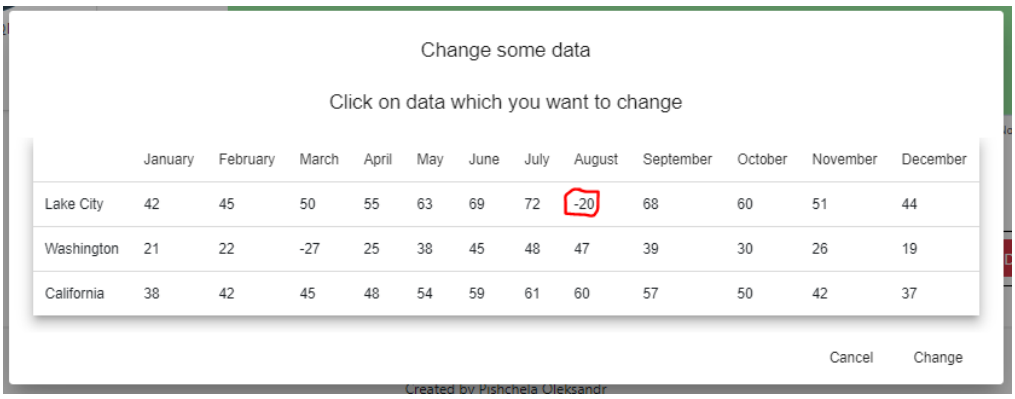


Рис. 5. Заміна деяких даних

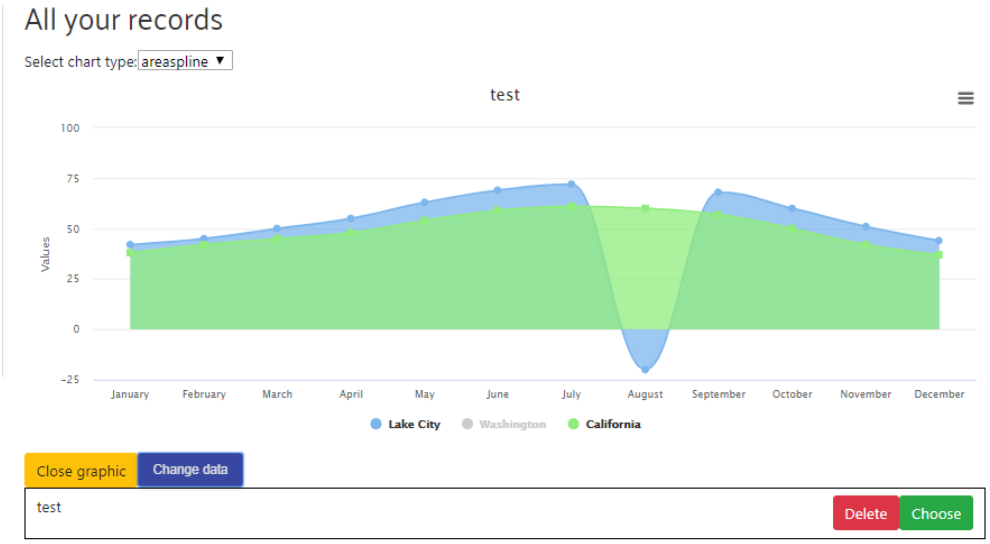


Рис. 6. Динамічно оновлені дані на графіку

У лівому верхньому кутку користувач має змогу вибрати тип представлення звіту на один із випадаючого списку (рис. 7).

Доступні типи графічних представлень:

– лінійний;

– закруглений лінійний;

– площа;

– закруглена площа;

– стовпці;

– розсіяні точки;

– бруски.

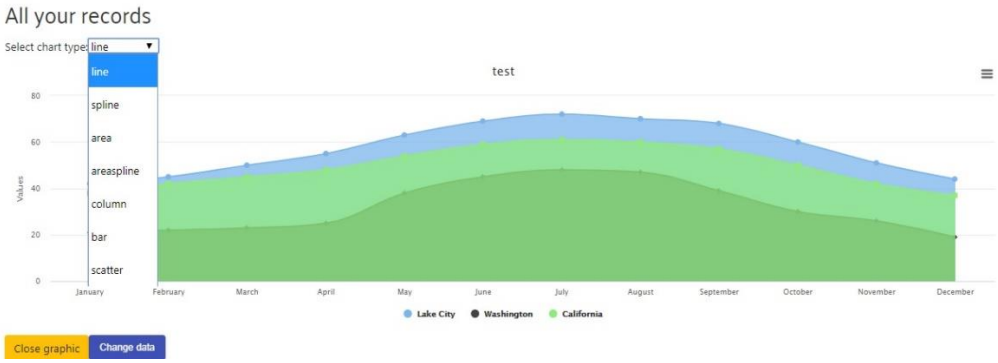


Рис. 7. Вибір типу відображення графіку з випадаючого списку

Натиснувши на кнопку у правому верхньому кутку (рис. 8) користувач має на вибір декілька функцій:

– розгорнути графік на весь екран;

– надрукувати графік (рис. 9);

– зберегти як PNG зображення;

– зберегти як JPEG зображення;

– зберегти як PDF документ;

– зберегти як векторне зображення SVG.

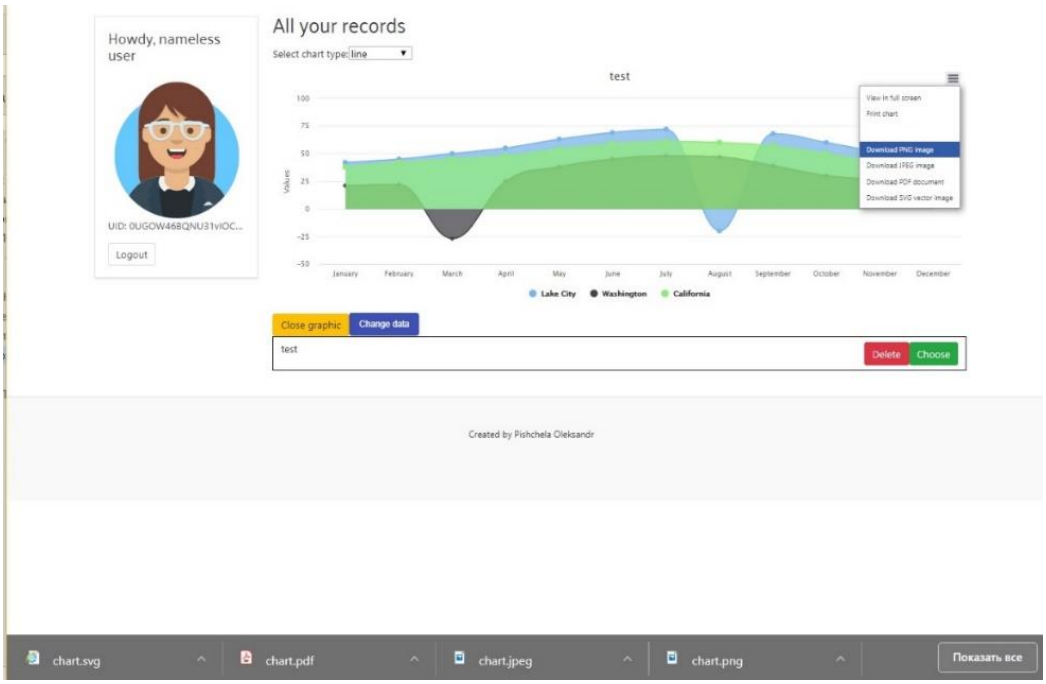


Рис. 8. Параметри збереження графіку

На рис. 8 знизу бачимо, що всі файли успішно завантажилися.

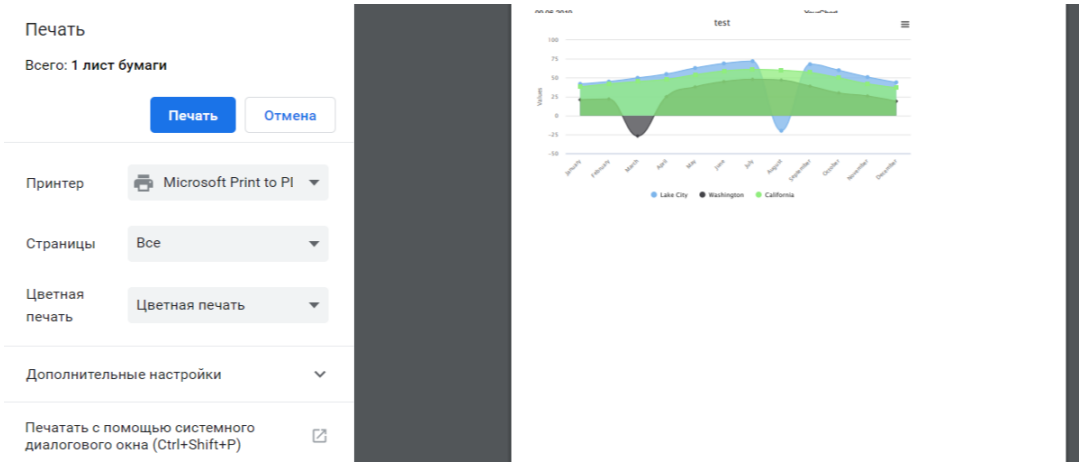


Рис. 9. Створений графік у вікні друку